UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Um Método de Inferência Lógica baseado na Transformação Dual

Dissertação submetida à Universidade Federal de Santa Catarina
como requisito parcial à obtenção do grau de

Mestre em Engenharia Elétrica

por

Isabel Tonin

Florianópolis, 06 de junho de 1997

Um Método de Inferência Lógica baseado na Transformação Dual

Isabel Tonin

Esta dissertação foi julgada para a obtenção do título de **Mestre em Engenharia**, modalidade **Engenharia Elétrica**, área de concentração **Sistemas de Controle, Automação e Informática Industrial**, e aprovada em sua forma final pelo curso de Pós-Graduação.

Florianópolis, 06 de junho de 1997

Prof. Guilherme Bittencourt, orientador

Prof. Adroaldo Raizer
coordenador do curso de Pós-Graduação em Engenharia Elétrica
da Universidade Federal de Santa Catarina.

Banca Examinadora

Prof. Guilherme Bittencourt, orientador

Prof. Arthur Buchsbaum

Prof. Janete Cardoso

Prof. Rosvelter J. Coelho da Costa

À Marcia,
minha irmã

"Não sei ... mas posso aprender!"

Marcelo Bittencourt

Agradecimentos

Agradeço, primeiramente, a todos os professores do LCMI pela acolhida no laboratório e, principalmente, ao prof. Edson de Pieri quem primeiro abriu suas portas.

À Capes pelo financiamento, que viabilizou a realização deste trabalho.

Ao prof. Rosvelter J. Coelho da Costa, pelo conhecimento, postura e confiança transmitidos em todas as etapas deste trabalho. A ele, meu especial agradecimento e consideração.

À Richard, Gerson, Nardenio, Ruben, Eliza, Rosimeire, Baiano, Cleto, Jezer e Max amigos e colegas, pelos momentos compartilhados.

Aos três mosqueteiros Cynara, Wail e Antonio, amigos inseparáveis, e também ao meu amigo Marcelo Fabian Oliva.

À minha mãe e ao meu irmão Leornardo, que me acompanharam de perto, pelo amor compreensivo e amigo.

Ao Raul, o último a fazer parte do percurso, pelo cuidado e apoio demonstrados em um período tão delicado. Pela sua dedicação e companheirismo, serei sempre grata.

Aos demais amigos e familiares que, de uma forma ou de outra, compartilharam e contribuiram durante este período.

Finalmente, ao meu orientador, Guilherme Bittencourt por tornar esta estrada tão agradável e proveitosa. Como agradecer? "Agradecimento inteligente pode tornar-se artificial", diria ele. Valeu, Gui!

Resumo

Este trabalho situa-se na área de Prova Automática de Teoremas (Inteligência Artificial), e baseia-se em um método de inferência correto e completo para a lógica de primeira ordem denominado Método de Inferência baseado na Transformação Dual. Este método é baseado na transformação de uma fórmula lógica escrita na forma normal canônica conjuntiva para a forma normal canônica disjuntiva e vice-versa. Este método explora a dualidade das possíveis representações de uma fórmula lógica, as formas normais conjuntiva e disjuntiva, e suas semânticas. O mesmo está baseado na transformação de uma forma normal para a outra. Embora aparentemente muito ineficiente, o método apresenta interessantes propriedades tais como não apresentar nenhuma regra de inferência externa ao sistema lógico.

A transformação entre as formas canônicas, isto é, a transformação dual, é uma das operações responsáveis pela ineficiência do método proposto. Para minimizar o custo desta operação, é proposto um algoritmo capaz de realizar esta transformação de forma eficiente. Este algoritmo possui, entre outras, duas importantes propriedades: gera diretamente o conjunto de implicantes primários de um conjunto de cláusulas ou cláusulas duais sem gerar cláusulas subsumidas e, por ter uma estrutura naturalmente concorrente, é facilmente paralelizável.

O principal objetivo deste trabalho é apresentar o método e o algoritmo propostos, passo a passo de forma a propiciar seu entendimento. Para tal, foi desenvolvido um ambiente que os implementa. Através deste foi possível depurar e estudar tanto o método quanto o algoritmo. As definições de rotinas e os exemplos apresentados neste trabalho baseiam-se nesta implementação.

Abstract

This work inscribes itself in the Automatic Theorem Proving domain (Artificial Intelligence). It is based on a correct and complete inference method for first-order logic, called Inference Method through Dual Transformation. This method is based on the dual transformation of a logic formula between the disjunctive and conjunctive canonical forms. Although apparently very inefficient, the method presents interesting properties, such as not to present any inference rules external from the logical system. The method just explores the duality of the possible representations of a logical formula and their semantics.

The transformation between canonical forms, i.e. the dual transformation, is one of the operations that cause the proposed method inefficiency. To minimize its cost, an efficient algorithm to perform it is proposed. This algorithm presents, among others, two important properties: it generates directly the prime implicants of a clause or dual clause set, without generating any subsumed clause and, because of its structure, it is naturally concurrent and therefore easy to parallelize.

The main goal of this work is to present the proposed inference method and dual transformation algorithm. To achieve this goal an environment was implemented that include them. Using this environment, it was possible to study, test and improve both the method and the algorithm. The procedure definitions and the proposed examples presented in this work are based on this implementation.

Sumário

1 Introdução 1

2 Inferência por Transformação Dual 7
2.1 Lógica Simbólica - Aspecto Dedutivo 7
2.1.1 Inferência 8
2.1.2 Regra de Inferência 10
2.2 Métodos de Inferência 11
2.2.1 Método de Prova Dedutivo 11
2.2.2 Método de Prova por Refutação 15
2.3 Método de Prova Automática 17
2.3.1 A Prova como Solução de Problemas 17
2.3.2 Forma Normal Conjuntiva e Forma Normal Disjuntiva 18
2.3.3 Subsunção 23
2.4 O Método por Transformação Dual 25
2.5 Conclusão 32

3 Um Algoritmo para Transformação Dual 34
3.1 A Transformação Dual 34
3.2 O Algoritmo de Transformação Dual 36
3.2.1 Dinâmica do Algoritmo 37
3.2.2 O Algoritmo 42
3.3 Conclusão 52

4 A Implementação 54
4.1 O LOGIK 54
4.2 A Interação com o Ambiente 55

5 Conclusão 67
5.1 O trabalho apresentado 67

5.2 PERSPECTIVAS 68

Lista de Figuras

2.1 Uma tira de quadrinhos de Calvin e Haroldo, de Waterson 12
2.2 Árvore de prova do exemplo 3 obtida por modus ponens 13
2.3 Árvore de prova do exemplo 3 obtida por silogismo hipotético e modus ponens 13
2.4 Árvore de prova por refutação do exemplo 5 18
2.5 Execução da rotina Pulsar para o exemplo 13 32

3.1 Predicados dos exemplos 6 e 7 no hipercubo 38
3.2 Cláusulas duais do exemplo 7 em seus vértices associados 39
3.3 Literais das cláusulas do exemplo 7 em seus vértices associados 39
3.4 Vértices com as combinações de literais e cláusulas duais 40
3.5 Ordem parcial para cláusulas duais do exemplo 7 42
3.6 Holograma associado ao exemplo 18 49

Lista de Tabelas

3.1 Cláusulas do exemplo 18 e quanta associados 45
3.2 Informações das órbitas no processamento local 50

Capítulo 1

INTRODUÇÃO

Métodos de Inferência Lógica despertam, desde há muito, bastante interesse na comunidade de Inteligência Artificial (IA) e, mais especificamente, na comunidade da IA Simbólica. Seu estudo define a área de pesquisa conhecida como *Prova Automática de Teoremas* que tem se mantido bastante ativa. A fim de melhor introduzir este trabalho, inicialmente discorremos sobre alguns pontos relativos à área[1]:

- *Inteligência Artificial?!*

  Qual a primeira idéia que nos vem à mente quando ouvimos falar de Inteligência Artificial? Robôs ... ou pelo menos máquinas pensantes. Nos lembramos dos filmes de ficção científica, com seus computadores capazes de responder (e até decidir) qualquer coisa, todo o tipo de maquinaria, eletrodomésticos e outras tantas traquitanas fazendo tudo automaticamente, robôs andando de um lado para outro, comunicando-se e até afeiçoando-se aos seres humanos. Quem não se lembra da dupla C3P0 e R2D2, leais ao seu dono Lucky Skywalker no filme *Guerra nas Estrelas*? O corpo era de lata, mas suas reações: alegres, preocupados, com medo em certas horas, atrapalhados... Ou ainda, mais "real", o assassino implacável quase humano enviado do futuro pelas "forças inimigas" no filme *O Exterminador do Futuro I* e que, na continuação do mesmo filme, retorna também do futuro, mas desta vez enviado pelas "forças do bem", passando a desempenhar o papel de herói da estória? Neste último, pelo menos, é passada a idéia de um ser programável, ou seja, que existe alguém que define o papel que este "humanóide" deve desempenhar. Teríamos ainda, como exemplo de "seres artificiais inteligentes", HAL no filme *2001-Uma Odisséia no Espaço*, não um

[1] Os conceitos apresentados estão baseados em [Bit96b], [Isr94] e [Pen91]

robô mas um computador que controla tudo, sabe de tudo e responde tudo. Mas vamos parar por aqui não dizendo que isto tudo seja uma bobagem, mas que estamos longe de tal realidade.

A possibilidade de criar uma "inteligência artificial" parece fascinar o ser humano desde há muito. De fato, a IA foi contruída a partir de idéias filosóficas, científicas e tecnológicas herdadas de outras ciências, algumas tão antigas como a lógica, com seus 23 séculos. Questões filosóficos como o chamado *problema corpo-mente*, motivam e dividem opiniões de pesquisadores da IA. Porém, oficialmente, enquanto ciência, a IA tem uma história de pouco mais de quarenta anos, iniciados com muita polêmica, expectativas exageradas e conseqüentes fracassos. Definições básicas como seu nome, seus objetivos, seu objeto de estudo ou o que vem a ser exatamente *inteligência* até hoje não obtiveram consenso na comunidade de IA.

Independentemente de definição formal precisa, o objetivo central da IA é imitar, através do uso de computadores, o máximo possível da atividade mental humana. As duas principais áreas de conhecimento que levaram à IA são a Lógica Matemática e a Computação, mas seus estudos apresentam hoje intersecção com outras grandes áreas de conhecimento como a filosofia, psicologia, biologia, entre outras.

Sua evolução deu-se em duas linhas distintas de pesquisa: a linha *Conexionista*, que visa a modelagem da inteligência humana através da simulação dos componentes do cérebro humano e que deu origem às Redes Neuronais Artificiais; e a linha *Simbólica*, cujos princípios são apresentados no artigo "Physical Symbol Systems" (Newell, 1980), e segue a tradição da lógica, tentando simular a inteligência humana enquanto comportamento lingüístico e simbólico. Observa-se hoje o crescimento de uma nova linha da IA, a *IA Evolutiva*, baseada na observação de mecanismos evolutivos encontrados na natureza, como auto-organização e comportamento adaptativo (observados nos modelos de algoritmos genéticos e autômatos celulares, por exemplo).

Cada uma destas linhas de pesquisa depara-se com limitações teóricas e tecnológicas, vantagens e desvantagens, oferecendo muitos desafios à pesquisa atual. Muito se tem avançado e estudado, porém embora muitas coisas inteligentes tenham sido feitas, a simulação de algo que pudesse passar por inteligência autêntica está ainda a uma enorme distância.

- *Símbolos e IA, um casamento lógico*

Para muitos, a lógica é a "matemática" da IA. A lógica pode ser vista como o estudo do raciocínio, mais propriamente como o estudo do raciocínio válido. Podemos também ver este raciocínio válido como inteligência. Assim sendo, a busca da IA por "dispositivos" capazes de apresentar um comportamento inteligente pode ser vista como a busca por dispositivos capazes de apresentar um comportamento que requer um raciocínio válido. Assim, o estudo do "raciocínio válido" é o ponto de contato entre estas duas áreas.

O uso da lógica simbólica na IA foi proposto inicialmente por P. Hayes, seguido por J. McCarthy e tantos outros. Eles observaram que programas de IA tinham de manipular uma grande quantidade de conhecimento que deveria, de alguma forma, estar representada nestes programas. Pareceu-lhes bastante apropriado representar formalmente este conhecimento independentemente do programa. A lógica clássica de primeira ordem era uma boa linguagem para isto, pois ela possui uma estreita relação com a linguagem natural, sua semântica é clara e bem estudada, e além disto apresenta um mecanismo formal para tornar explícito o conhecimento implícito em uma dada formalização: a dedução lógica.

Inicialmente, a pesquisa na IA Simbólica concentrou-se na busca de formalismos gerais capazes de resolver qualquer tipo de problema; pouco depois viu-se que tal rumo de pesquisas era inviável, pois esbarrava em dois sérios problemas:

- o primeiro problema é teórico e advém do uso de modelos baseados em lógica de primeira ordem como formalismo básico. É conhecido como "Explosão Combinatória" e é inerente aos métodos baseados em lógica: a memória e o tempo necessários para a resolução de um problema cresce exponencialmente com o tamanho do mesmo;
- o segundo recai sobre o fato de ser necessário, para resolver qualquer tipo de problema, a representação do conhecimento total do mundo real - inclusive conhecimentos de senso comum do comportamento humano. Estes são normalmente de dfícil manipulação pela lógica de primeira ordem por serem incompletos, ou parcialmente incoerentes. Mesmo assim, a lógica clássica pode ser adaptada e estendida para representar diversos tipos de problemas mais específicos (garantindo uma certa eficiência na solução de problemas, mesmo que resultando em lógicas "mais fracas" que a lógica clássica de primeira ordem).

Muitas soluções foram e continuam sendo propostas e, apesar dos problemas, a lógica tem desempenhado dois importante papéis na IA:

- ser fonte de linguagens e "lógicas" para o desenvolvimento de programas inteligentes.
- fornecer subsídio matemático e conceitual subjacente às várias áreas de pesquisa em IA, como Representação de Conhecimento, Prova Automática de Teoremas e Sistemas Especialistas por exemplo.

Dois importantes resultados são a base do poder da lógica e também de suas limitações: o primeiro, provado independentemente em 1930 por Kurt Gödel e Jacques Herbrand, afirma que existem sistemas lógicos de prova, ou métodos de prova, que são *completos*, ou seja nos quais toda fórmula verdadeira pode ser provada. O segundo, provado em 1936, também independentemente por Alonzo Church e Alan Turing, afirma que a lógica é *semi-decidível*, isto é, não existe um método geral capaz de decidir, em um número finito de passos, se uma fórmula é verdadeira. Isto significa que, dada uma fórmula verdadeira é sempre possível encontrar uma prova, mas que existem fórmulas não verdadeiras para as quais não existe nenhum método finito que, garantidamente, seja capaz de decicir sobre sua não validade (então, métodos para a prova automática de teoremas podem nunca parar).

Este resultado, um tanto negativo, da lógica é um dos grandes argumentos dos críticos da IA simbólica. Já, para seus adeptos, apesar de extremamente importante, ele não impossibilita que, utilizando a lógica, as máquinas um dia estejam aptas a raciocinar quase tão bem quanto pessoas. Justificam que, a despeito da incompletude de alguns sistemas, seres humanos conseguem utilizá-los de maneira bastante razoável.

- *Lógica* $\cap$ *IA* = *Raciocínio Válido*

Raciocínio válido é, de certa forma, um raciocínio formal, que segue algumas regras pré-estabelecidas. Ao falar-se em "raciocínio formal" pode parecer que está-se tentando criar regras para raciocinar através delas. Porém, a idéia é examente oposta. Pensa-se, ou raciocina-se, simplesmente. O papel da lógica é tentar modelar formalmente este raciocínio. Sua história remonta aos antigos filósofos gregos, principalmente Aristóteles, que estabeleceu seus fundamentos de maneira sistemática.

Este raciocínio, ao ser "formalizado", define *métodos de inferência lógica*, ou seja, métodos formais de dedução lógica. Através destes métodos, conclusões podem ser obtidas, baseadas em premissas. Sendo as premissas vistas como conhecimentos, pode-se afirmar que, nelas baseados, novos conhecimentos são explicitados através dos métodos de inferência lógica. Estes métodos de inferência são também chamados de *métodos de prova (ou métodos de prova de teoremas)*. Os primeiros métodos de prova foram concebidos para serem utilizados por seres humanos. A partir da introdução por Robinson [Rob65] e Smullyan [Smu68], em 1960, de métodos eficientes para prova automática de teoremas por computador, a lógica passou a ser estudada também como método computacional para a solução de problemas.

O presente trabalho visa apresentar um método de inferência proposto no artigo *Concurrent Inference through Dual Transformation* [Bit97a] denominado *Método de Inferência baseado na Transformação Dual*. Sua apresentação dá-se de maneira informal, preocupando-se mais em explicar o funcionamento e propriedades do mesmo do que apresentar sua formalização (que pode ser encontrada no artigo onde o mesmo é proposto). Este trabalho apresenta também um algoritmo para a transformação dual propriamente dita proposto no mesmo artigo e em [TB97]. Para tal, o trabalho está assim organizado:

- Capítulo 2
  São abordados os aspectos dedutivos da lógica e as características gerais dos métodos de prova de teoremas, bem como o embasamento lógico necessário ao entendimento do trabalho. Dados os fundamentos é apresentado o método proposto.

- Capítulo 3
  É colocado o problema da transformação entre as formas normais canônicas conjuntiva e disjuntiva da lógica de primeira ordem. Passa-se à apresentação intuitiva da dinâmica do algoritmo proposto para a transformação dual. Sua definição dá-se na seqüência.

- Capítulo 4
  É apresentada, em linhas gerais, a implementação do ambiente, composto pelo método e algoritmo propostos. São apresentados e discutidos exemplos de execuções do algoritmo.

- Capítulo 5
  São apresentadas conclusões do trabalho realizado, estado atual da pesquisa e perspectivas futuras.

Capítulo 2

Inferência por Transformação Dual

Neste capítulo é apresentado um método de inferência para lógica de primeira ordem denominado *Inferência Concorrente por Transformação Dual* (ou simplesmente Inferência por Transformação Dual).

O método proposto é um método correto e completo de inferência para lógica de primeira ordem baseado na transformação entre as formas normal conjuntiva e disjuntiva de uma fórmula lógica, a *transformação dual.* Sua definição difere um pouco da originalmente proposta, devido a alguns avanços teóricos que ocorreram ao longo do desenvolvimento deste trabalho.

Antes de apresentar o método, são apresentados alguns mecanismos dedutivos da lógica simbólica. São apresentadas também, de uma forma geral, as características de métodos de inferência lógica. Estas noções são apresentadas a fim de familiarizar o leitor com os termos e conceitos necessários ao entendimento do método proposto.

Em seguida, apresenta-se o Método de Inferência por Transformação Dual propriamente dito, e encerra-se comentando algumas de suas características.

2.1 Lógica Simbólica - Aspecto Dedutivo

A Lógica moderna data de 1879 quando Frege publicou a primeira versão do que hoje é conhecido como *Cálculo de Predicados.* No final do século XIX a lógica passou a ser utilizada como base formal para outros campos da matemática, e seus limites passaram a ser estudados com rigor por teóricos como David Hilbert, Giuseppe Peano, Georg Cantor e Thoralf Skolem. A lógica pode ser vista como o estudo do raciocínio, mais precisamente o estudo do raciocínio válido. A lógica busca formalizar um raciocínio

válido tornando-o sistemático. Para tal, ela trata fundamentalmente de dois conceitos: *Verdade* e *Prova*. Estes conceitos foram investigados durante séculos por filósofos, matemáticos e lingüistas.

A parte da lógica que estuda os *valores verdade* (do inglês "truth values") é chamada *Teoria dos Modelos*. Neste trabalho são apresentados apenas alguns conceitos desta teoria. As frases em lógica são chamadas de fórmulas. As fórmulas constituem uma linguagem formal. Cada fórmula pode ser associada a um *valor verdade* isto é, ao valor verdade *verdadeiro (V)* ou ao valor verdade *falso (F)*. Uma maneira formal de atribuição de valores verdade a fórmulas foi proposta por Alfred Tarski em 1934 e ficou conhecida como *Semântica de Tarski*. Esta atribuição é chamada de *interpretação*. Uma interpretação que torna uma fórmula verdadeira, isto é, que *satisfaz* uma fórmula, é chamada de *modelo*. Uma fórmula é dita *satisfazível* (do inglês "satisfiable") ou *consistente* se ela possui pelo menos um modelo. Se toda interpretação possível for um modelo para a fórmula, esta é dita *válida* ou *tautológica*. Por outro lado, uma fórmula que é tornada falsa em pelo menos uma interpretação é dita *inválida* e, se ela é tornada falsa por toda interpretação possível (não possui nenhum modelo) é dita *insatisfazível* (do inglês "unsatisfiable") ou *contraditória* [CRT73].

A parte da lógica que estuda as provas é chamada *Teoria das Provas*. Nela, é explorada a estrutura dedutiva da linguagem lógica, seus mecanismos de dedução (inferência) de novas fórmulas. A seqüência de fórmulas geradas a partir um conjunto inicial de fórmulas é chamada de *prova*. Um *teorema* é uma fórmula válida. Provar um teorema δ é demonstrar a sua validade, normalmente demonstrando que o mesmo pode ser deduzido a partir de (ou é implicado por) um conjunto inicial de fórmulas Δ, ou seja, que a fórmula $\Delta \rightarrow \delta$ é uma tautologia ($\rightarrow$ denota a implicação). Esta parte da lógica será apresentada com mais detalhes no decorrer deste capítulo.

2.1.1 INFERÊNCIA

A *dedução* ou *inferência lógica* (ou simplesmente inferência) é um mecanismo formal para geração de conclusões a partir de premissas. Ou seja, baseando-se apenas em premissas, supostas verdadeiras, fatos podem ser concluídos também como verdadeiros. As conclusões obtidas não alteram o valor verdade do conjunto de premissas, isto é, sendo as premissas verdadeiras, as conclusões também o são.

Exemplo 1 *Qualquer leitor das tiras de quadrinhos de Waterson intitulada "Calvin e Haroldo" sabe que (i) quando a mãe de Calvin está muito irada é porque ele aprontou "uma das suas". Conhecendo este fato, o que se pode concluir vendo um quadrinho*

onde (ii) a mãe de Calvin está muito irada? Utilizando-se os conhecimentos (i) e (ii) como premissas pode-se deduzir que (iii) Calvin aprontou uma das suas.

Nota-se que a partir dos fatos já conhecidos (i) e (ii), pela inferência, foi gerado um novo conhecimento (iii); não novo no sentido de que não existia anteriormente, mas tornado explicitamente conhecido, pois o mesmo já era implicitamente "expresso" pelas premissas. Esta habilidade de aquisição de novos conhecimentos pela realização deste tipo de raciocínio (inferência) é uma parte essencial da inteligência [GN87].

O raciocínio empregado no exemplo anterior é bastante simples: acreditar em uma premissa é considerá-la verdadeira; acreditar na premissa (i) não significa afirmar que a mãe de Calvin esteja irada, mas afirmar que "sempre" que ela está irada ele aprontou uma das suas; por sua vez, acreditar na premissa (ii) é afirmar que realmente (é verdadeiro que) a mãe está irada; portanto, para acreditar nas duas premissas ao mesmo tempo, *tem-se* de acreditar na conclusão (iii). Este raciocínio é uma *inferência lógica.* A conclusão obtida é uma *conclusão lógica.* Diz-se que a conclusão *segue logicamente* do conjunto de premissas, ou aquela é *logicamente implicada* por este. É bom ter cuidado com o significado da premissa (i) deste exemplo. Ela expressa uma *implicação* entre a ira da mãe e as "artes" de Calvin. *Se* a mãe está irada *então* Calvin aprontou e não o contrário. Então, se a premissa (ii) deste exemplo fosse *Calvin aprontou uma das suas* não poderia ser deduzido que sua mãe está irada (ela pode não ter ficado sabendo).

Exemplo 2 *Porém, se ao invés das premissas do exemplo 1 descreverem a relação entre as artes de Calvin e a ira de sua mãe, elas descrevessem a natureza imaginária das conversas de Calvin com Haroldo, seu tigre de pelúcia. Elas poderiam expressar que (i) se Haroldo é um tigre de pelúcia, suas "falas" são imaginárias e (ii) Haroldo é um tigre de pelúcia, uma conclusão logicamente inferida seria (iii) suas falas são imaginárias.*
Para a premissa (i) deste novo exemplo ser verdadeira, Haroldo pode ou não ser um tigre de pelúcia. Mas se o for, dizer que suas falas são imaginárias expressa uma verdade.

Novamente acreditando-se nas duas premissas, acredita-se também na conclusão, ou seja, a conclusão é logicamente implicada pelas premissas. Nos dois exemplos o raciocínio necessário para se alcançar a conclusão é o mesmo. De fato, a estrutura das sentenças (premissas e conclusões) que os compõem é a mesma, o que os diferencia é o significado que cada termo da linguagem utilizada possui, dando a cada exemplo uma interpretação própria. Uma interpretação representa um significado atribuído a uma fórmula lógica. Cada exemplo pode ser visto como uma *interpretação* para a mesma

classe (ou padrão) de sentenças que têm como premissas (i) $\alpha \rightarrow \beta$ e (ii) α, e como conclusão (iii) β. No exemplo 1 a interpretação de α é "a mãe de Calvin está muito irada" e a interpretação de β é "Calvin aprontou uma das suas". Já no exemplo 2, a interpretação de α é "Haroldo é um tigre de pelúcia" e a de β é "as falas de Haroldo são imaginárias".

Este tipo de raciocínio, utilizado para gerar a conclusão, independe da interpretação. Sabe-se no entanto que, baseados nas duas premissas padrão, a conclusão obtida é uma conclusão lógica. O raciocínio em questão expressa uma verdade lógica, estrutural, uma *tautologia* — verdade lógica que se mantém sob qualquer interpretação. Pode-se atribuir qualquer significado para α e β que, ao acreditar-se nas premissas está-se, também, acreditando na conclusão. "Logicamente" falando, este é um *raciocínio válido.*

Ao modelar um raciocínio, sentenças lógicas são utilizadas como premissas e conclusões. Estas sentenças são representadas em símbolos lógicos de acordo com uma sintaxe pré-estabelecida pela linguagem lógica. Porém ao representar-se informações em fórmulas lógicas, normalmente tem-se em mente uma interpretação específica para elas. Foram mostradas nos exemplos anteriores duas situações distintas que podem ser representadas exatamente da mesma forma. Além destas, existe uma infinidade de outras interpretações possíveis para as mesmas fórmulas das sentenças. Como garantir que uma conclusão obtida é uma conclusão verdadeira para a interpretação desejada? Uma forma é vincular o valor verdade da conclusão ao valor verdade das premissas. Pode-se, por garantia, obter-se apenas aquelas conclusões que são verdadeiras simultaneamente com as premissas, isto é, aquelas que têm os mesmos modelos que as premissas.

2.1.2 REGRA DE INFERÊNCIA

A regra formal que modela (representa) um passo de um raciocínio é chamada de *regra de inferência.* Uma regra de inferência é uma forma bem definida de manipular as premissas sintaticamente gerando uma conclusão. Sintaticamente porque a regra torna explícitas as tautologias expressas na estrutura das premissas, em sua escrita (e não em seu significado). Uma regra de inferência consiste de (i) um conjunto de sentenças padrão chamado de *condições* e (ii) um outro conjunto de sentenças padrão chamado de *conclusões.* Sempre que as premissas "casam" com as condições está habilitada a geração de sentenças que "casam" com as conclusões.

De maneira geral, um sistema lógico consiste em um conjunto de fórmulas (as premissas) e um conjunto de regras de inferência. Existem várias regras definidas, e cada uma delas modela um raciocínio correto. Por exemplo, uma regra de inferência bastante conhecida é a regra *Modus Ponens* que diz que, sempre que acreditamos que

(i) uma sentença α implica outra sentença β e também acreditamos (ii) na sentença α, temos de acreditar também (iii) na sentença β. Os ítens (i) e (ii) são as condições padrão e o item (iii) é a conclusão padrão.

Para ser utilizada, uma regra de inferência deve ser necessariamente *correta* (do inglês "sound"), ou seja, se aplicada a premissas verdadeiras a conclusão gerada tem de ser verdadeira também. Este conceito é a base da *implicação lógica.* A conclusão é uma *conseqüência lógica* das premissas, é implicada por estas. Uma conseqüência lógica é verdadeira em todas as interpretações nas quais suas premissas o forem (ou seja, toda interpretação que é um modelo para as premissas também o é para suas conseqüências lógicas). Isto quer dizer que se um conjunto de premissas é satisfazível, o conjunto resultante de sua união com uma de suas conseqüências lógicas também é satisfazível.

2.2 MÉTODOS DE INFERÊNCIA

Em 1930, Kurt Gödel e Jacques Herbrand provaram, independentemente, que existem sistemas lógicos de prova, ou *métodos de prova*, que são *completos* (do inglês "complete"), ou seja, nos quais toda fórmula verdadeira pode ser provada. Estes métodos são procedimentos formais para provar/gerar/explicitar teoremas, e também são conhecidos como *Métodos de Inferência.*

2.2.1 MÉTODO DE PROVA DEDUTIVO

De um modo geral, para provar um teorema a partir de premissas, um método de prova gera os teoremas por elas implicados. Se o teorema desejado não foi gerado (nem a sua negação), os teoremas gerados são unidos às premissas e o ciclo é recomeçado. Isto se repete até que o teorema desejado (ou sua negação) seja gerado, ou seja, até sua prova ser alcançada.

Exemplo 3 *Voltando ao "Calvin e Haroldo", vê-se em uma de suas tiras, ilustrada na figura 2.1, uma seqüencia de cenas onde ambos estão dançando freneticamente. O último quadro mostra um quarto escuro com a mãe de Calvin sentada na cama, recém acordada, comentando com o marido: "ou eu ainda estou sonhando ou ele está tocando música clássica a 78 rpm". Neste caso, pode-se querer provar o óbvio: Calvin está realmente tocando música clássica a 78 rpm.*

Como premissas têm-se: (i) Se ela (mãe) não está sonhando ele (Calvin) está tocando música clássica a 78 rpm , (ii) Se ela está acordada ela não está sonhando e

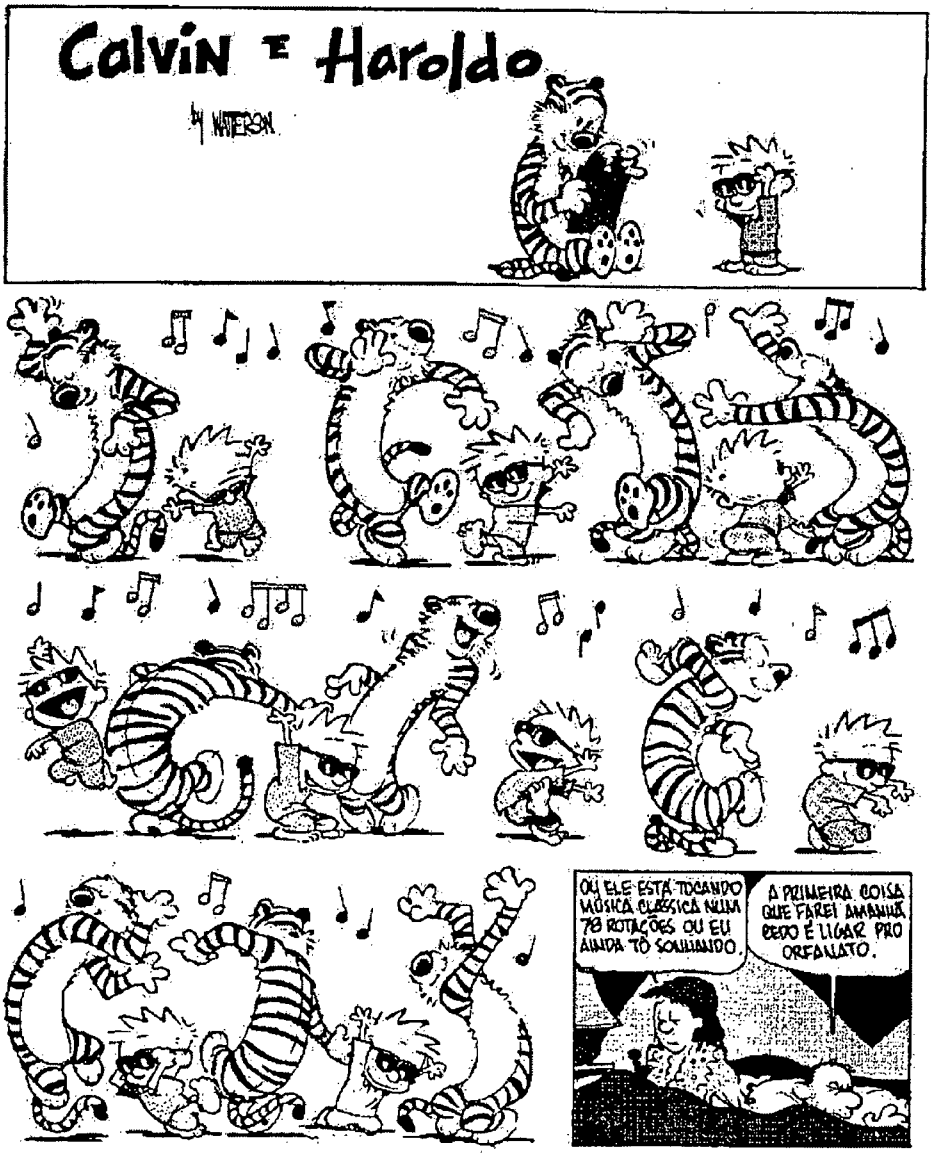

Figura 2.1: Uma tira de quadrinhos de Calvin e Haroldo, de Waterson

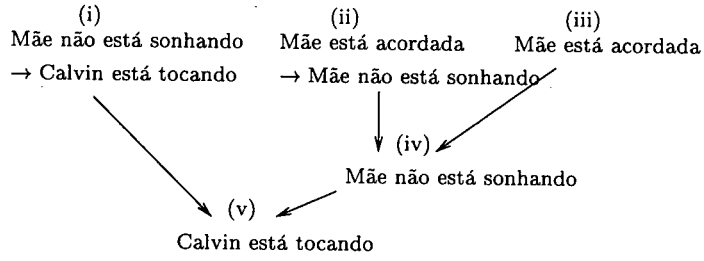

Figura 2.2: Árvore de prova do exemplo 3 obtida por modus ponens

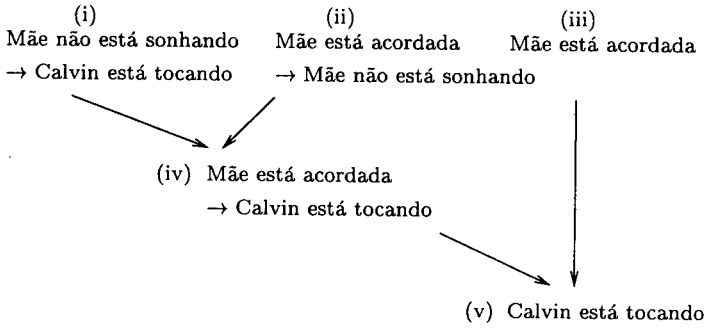

Figura 2.3: Árvore de prova do exemplo 3 obtida por silogismo hipotético e modus ponens

(iii) ela está acordada. Pelas premissas (ii) e (iii) em um primeiro passo conclui-se que (iv) ela não está sonhando. Adotando-se (iv) como premissa e a premissa inicial (i) pode-se concluir que (v) ele está tocando música clássica a 78 rpm, como era de se esperar.

A prova foi alcançada em dois passos. A seqüencia de deduções lógicas realizadas para alcançar o teorema ((ii) e (iii), (iv) e (i), (v)) é a sua prova. Esta seqüência também é chamada de Árvore de Prova e pode ser visualizada na figura 2.2.

Por outro lado, pelas premissas (i) e (ii) vemos que uma "cadeia" de implicações é formada. Esta cadeia pode ser resumida, e ser concluído que (iv) Se ela está acordada então ele está tocando música clássica a 78 rpm. Este "resumo" de uma cadeia de implicação expressa a transitividade da implicação e é denominada Silogismo Hipotético. *A conclusão final (v) "Calvin está tocando música clássica a 78 rpm" pode, então, ser obtida pelas premissas (iii) e (iv).*

A prova neste caso, apesar de representar um caminho diferente de deduções ((i) e (ii), (iii) e (iv), (v)), também foi obtida em dois passos. A árvore desta prova é mostrada na figura 2.3.

Neste caso a prova foi alcançada quase que diretamente. Em problemas um pouco mais complexos, porém, percebe-se que, ao serem gerados todos os teoremas em cada passo, muitos deles são "ruins" no sentido que são desnecessários para a prova. A busca pelos "bons" teoremas define *estratégias de prova.* Esta estratégias "guiam" a prova, evitando assim o crescimento exponencial do espaço de busca (número de possibilidades a serem avaliadas a cada passo). Ainda hoje mantém-se bastante ativa a pesquisa de estratégias de prova, mesmo para métodos já clássicos como, por exemplo, o método da resolução introduzido por Robinson em 1960.

Para formalizar os conceitos abordados, vejamos as seguintes relações: se toda interpretação que satisfaz todas as fórmulas de um conjunto de fórmulas G, simultaneamente, satisfaz também uma fórmula W, diz-se que W é uma *Conseqüência Lógica* de G. Esta relação entre as fórmulas é notada por:

$$G \models W$$

Se, através de um método de prova, ou seja, se, através de uma seqüência de aplicações de regras de inferência, a fórmula W pode ser obtida a partir de G, diz-se que W pode ser provada a partir de G, e esta relação é notada por:

$$G \vdash W$$

Caso o método de prova utilizado na prova de W seja correto, isto é, contenha apenas regras de inferência corretas, então: se W pode ser provada a partir de G, W é uma conseqüência lógica de G. Esta relação é notada por:

$$G \vdash W \Rightarrow G \models W$$

Em contrapartida, se toda fórmula W, conseqüência lógica de G, pode ser provada a partir de G por meio de um método de prova, este é dito *completo*, e esta propriedade é notada por:

$$G \models W \Rightarrow G \vdash W$$

Então, em sistemas corretos e completos

$$G \vdash W \Leftrightarrow G \models W$$

as propriedades de *provabilidade* e *implicação lógica* são equivalentes.

Os primeiros métodos construtivos de prova corretos e completos foram criados por Gödel e Herbrand. Existem hoje diversos métodos de prova corretos e completos tendo

como precursores os *Sistemas de Axiomas, Dedução Natural e Seqüentes de Gentzen* [Fit90].

Este conceito de prova é muito importante para a IA porque ele é a base da automatização de um raciocínio lógico. A partir de um conjunto de premissas que definem um problema, de uma fórmula a ser provada e de um método de inferência, pode-se gerar conseqüências lógicas das premissas. Dentre estas, pode-se verificar se a fórmula a ser provada, ou sua negação, foi gerada. Se a própria fórmula foi gerada ela é um teorema. Se, ao invés, sua negação foi gerada, sua negação é um teorema e, por conseguinte, a fórmula não o é[1] (a menos que o conjunto de premissas seja contraditório, do qual uma fórmula lógica com valor verdade falso pode ser conseqüência lógica).

Para alguns conjuntos de premissas é garantido que é alcançada a prova de uma fórmula ou de sua negação. Infelizmente, isto não é verdade para todos os conjuntos de premissas. De fato, existem fórmulas que nem elas próprias nem suas negações são conseqüências lógicas do conjunto de premissas. Isto torna a lógica apenas *semi-decidível*, como foi provado independentemente por Alonzo Church e Alan Turing.

A semi-decidibilidade da lógica significa que, se uma fórmula é um teorema ela pode (com certeza) ser provada dentro do sistema lógico (e sua negação pode ser refutada, como será visto a seguir). No entanto, não existe um método geral capaz de decidir, em um número finito de passos, se uma fórmula é ou não uma conseqüência lógica do conjunto de premissas. Disto resulta que métodos de prova automática podem não terminar. Vários tópicos são encontrados na literatura relacionados a esta propriedade de decidibilidade, como por exemplo: o teorema de Gödel e a incompletude de qualquer axiomatização finita da aritmética, o problema da *parada* de Turing e o conceito de "computabilidade", a matemática não recursiva e a descrição de conjuntos não recursivos mas recursivamente enumeráveis (quando apenas os elementos do conjunto podem ser calculados algoritmicamente e não os elementos de seu complemento), o teorema de Herbrand e as árvores semânticas infinitas com fechamento finito.

2.2.2 MÉTODO DE PROVA POR REFUTAÇÃO

Viu-se que se pode provar um teorema demonstrando que o mesmo é uma conseqüência lógica de um conjunto inicial de premissas. Uma outra forma de prová-lo é demonstrar que sua negação não pode ser verdadeira, ou seja, é refutá-lo. Refutar uma fórmula é demonstrar que a mesma implica uma contradição, que é insatisfazível (ou seja, que *a sua negação* é um teorema).

[1] Pelo princípio do terceiro excluído (do latim *Tertium Non Datur*) da lógica clássica: se uma fórmula é verdadeira a sua negaçao é falsa e vice-versa.

O raciocínio subjacente aos métodos de refutação é simétrico ao subjacente aos métodos de prova dedutivos. Como antes, crer nas premissas é crer que elas possuem pelo menos uma interpretação que as satisfaça simultaneamente. Demonstrar que uma fórmula não é um teorema é mostrar que nenhuma interpretação que satisfaz as premissas satisfaz a fórmula. Novamente, como o número de interpretações pode ser infinito, mostra-se pelas contradições estruturais que nenhum modelo pode satisfazer fórmula e premissas ao mesmo tempo.

Então, refutar a fórmula W é demonstrar que o conjunto de fórmulas H resultante da união de $\neg W$ com o conjunto de premissas G

$$H = G \cup \{\neg W\}$$

é insatisfazível (contradição). Simetricamente: para *provar* (deduzir) um teorema os métodos valem-se das tautologias expressas pelas fórmulas e para *refutá-lo*, das contradições.

Exemplo 4 *Vejamos como seria refutar um teorema utilizando o exemplo 2. Dele pode-se querer provar que "as falas de Haroldo são imaginárias", sendo este então o teorema. Para refutá-lo, em primeiro lugar, tem-se de uni-lo negado às premissas. Com isto tem-se como premissas: (i) se Haroldo é um tigre de pelúcia, suas "falas" são imaginárias, e (ii) Haroldo é um tigre de pelúcia. A nova premissa (iii) suas falas não são imaginárias. Pelas premissas (i) e (ii) infere-se que (iv) suas falas são imaginárias o que contradiz a premissa (iii). Nenhuma interpretação pode satisfazer uma fórmula e sua negação ao mesmo tempo. Alcançada a contradição sabe-se, então, que a negação da premissa (iii) é realmente um teorema.*

Assim como a prova dedutiva, a prova por refutação é a seqüência de fórmulas geradas até a contradição ser alcançada. Valem para um método de refutação as mesmas noções de *correção* (do inglês "soundness") e *completude* (do inglês "completeness") apresentadas para os métodos de prova dedutiva. Os métodos de refutação mais conhecidos não são completos[2], mas *completos para a refutação*. Um método é completo para a refutação se, através dele, todo teorema pode ter a sua negação refutada.

[2]Nem toda fórmula verdadeira pode ser provada. Por exemplo, utilizando-se o método de resolução, que é um método de refutação, do conjunto de premissas {P(x)} não se pode provar a fórmula P(a) mas apenas refutar a sua negação,$\neg P(a)$

2.3 MÉTODO DE PROVA AUTOMÁTICA

Os primeiros métodos de prova desenvolvidos eram mais apropriados para serem utilizados por seres humanos. A implementação destes métodos em computadores mostra-se normalmente ineficiente, ou pela quantidade de regras de inferência disponíveis, ou devido ao fato das regras envolverem escolhas ou, ainda, pela complexidade das mesmas. No entanto, a partir da introdução por J. A. Robinson e R. M. Smullyan na década de 1960 dos métodos de *Resolução* e dos *Tableaux Semânticos* respectivamente, diversos procedimentos eficientes voltados para a prova automática de teoremas por computador têm sido propostos.

Nesta seção são abordadas algumas características comuns aos métodos de prova automática por computador.

2.3.1 A PROVA COMO SOLUÇÃO DE PROBLEMAS

Com a introdução de métodos eficientes para prova automática de teoremas por computador, a lógica passou a ser estudada também como método computacional para a solução de problemas. Até agora foram apresentados nos exemplos premissas e teoremas cuja única função da prova era demonstrar se o teorema era de fato um teorema ou não. Se vistas as premissas como a descrição de um fato e o teorema como uma pergunta, esta seria (até agora) respondida com sim ou não. Porém, utilizar um método de inferência para solução de problemas é mais que isto. É poder extrair respostas para questões desconhecidas. Este uso para solução de problemas é mostrado no exemplo 5.

Exemplo 5 *Assumindo-se que toda pessoa gosta de seus amigos, e que Calvin gosta de Haroldo e de Susie pode-se querer saber* **quem** *são os amigos de Calvin. Não é o mesmo que querer saber se Susie ou Haroldo são amigos de Calvin. As premissas do problema são: (i) se x gosta de y então x é amigo de y, (ii) Calvin gosta de Haroldo e (iii) Calvin gosta de Susie. E o teorema? A resposta que espera-se é Calvin é amigo de fulano, então o teorema é Calvin é amigo de z, que negado fica (iv) Calvin não é amigo de z. Este exemplo apresenta duas refutações possíveis sendo, cada uma delas, uma resposta para o problema. As refutações são apresentadas na figura 2.4. A resposta da pergunta "Quem são os amigos de Calvin?" são dadas pelos valores que a variável z do teorema pode assumir, no caso Haroldo e Susie.*

Neste exemplo nota-se que, para expressar relações gerais como a da premissa (i), precisou-se adotar o uso de variáveis, no caso, x e y. Também no teorema, precisou-se utilizar uma variável, no caso, z. As variáveis são como lacunas que podem ser preenchidas com diversos valores. Ao expressar o teorema com uma variável, quer-se na

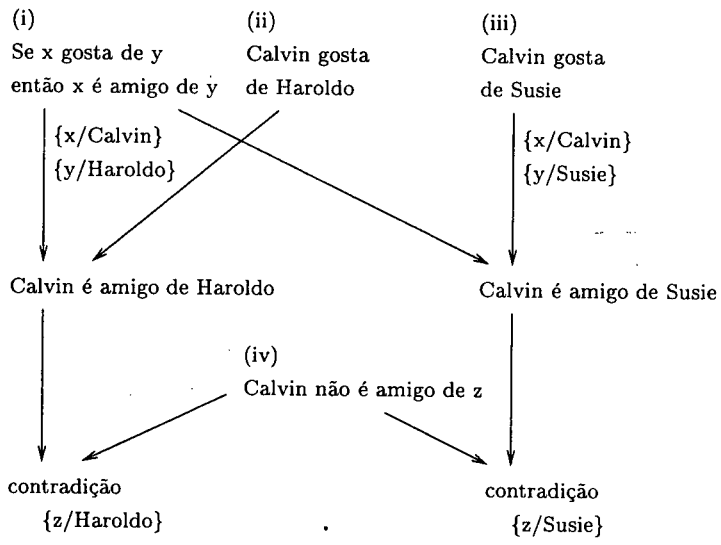

Figura 2.4: Árvore de prova por refutação do exemplo 5

realidade, saber que valores a ela podem ser atribuídos para tornar o teorema verdadeiro. O preenchimento destas lacunas é dado pela substituição de toda ocorrência de uma variável por um termo[3]. Esta troca de toda ocorrência de uma variável pelo termo definido na substituição é chamada de *aplicação da substituição.* As substituições aplicadas para a refutação são mostradas na figura 2.4 (notadas pelo par {variável/termo}).

2.3.2 FORMA NORMAL CONJUNTIVA E FORMA NORMAL DISJUNTIVA

Na definição da sintaxe da linguagem lógica encontra-se a definição dos conectivos (operadores binários) **e** ($\wedge$), **ou** ($\vee$) e **implica** ($\rightarrow$), e do operador unário de **negação** ($\neg$). Existem diversos subconjuntos destes operadores que são completos no sentido de poderem expressar qualquer fórmula lógica, por exemplo, $\{\neg, \rightarrow\}$, $\{\neg, \wedge\}$, $\{\neg, \vee\}$ e $\{\neg, \wedge, \vee\}$. Qualquer fórmula lógica pode ser reescrita para qualquer um destes subconjuntos de operadores.

Uma fórmula lógica qualquer pode ser representada em uma forma padrão bem definida, isto é, em *forma normal canônica.* Estas formas normais, além de padronizarem a fórmula, simplificam-na pelo uso restrito de operadores. Na busca de maior eficiência, a maioria dos métodos de prova voltados para a prova automática de teoremas baseia-

[3]Um termo representa um elemento do domínio e pode ser: um símbolo de variável, de constante ou de função enária tendo como parâmetros n termos.

se nestas formas normais, as chamadas *Forma Normal Conjuntiva* (ou Forma Clausal) e *Forma Normal Disjuntiva* (ou Forma Clausal Dual). O uso destas formas normais impõe aos métodos nelas baseados diversas características comuns. Maiores detalhes destas características podem ser encontrados em [WV94].

Para apresentá-las é necessário antes apresentar dois conceitos: disjunção e conjunção. Uma *disjunção* de fórmulas lógicas é a conexão destas pelo operador lógico $\vee$ e é notada por $[\ldots]$. Já uma *conjunção* é a conexão de fórmulas lógicas pelo operador lógico $\wedge$ e é notada por $\langle\ldots\rangle$. Estas definições e notações podem ser encontradas em [Fit90].

Transformar um conjunto qualquer de fórmulas para a forma normal conjuntiva é transformá-lo em uma *conjunção de cláusulas.* Uma cláusula é uma *disjunção de literais*[4]. Então um conjunto de fórmulas está em forma normal disjuntiva, se ele está na forma $C_1 \wedge C_2 \wedge \ldots \wedge C_n$ onde cada C_i é uma cláusula. Uma fórmula é uma cláusula se está na forma $L_1 \vee L_2 \vee \ldots \vee L_n$, onde cada L_i é um literal.

Pelas definições acima, percebe-se que o subconjunto de operadores lógicos utilizados na forma normal conjuntiva é $\{\neg, \wedge, \vee\}$. Como os conectivos "$\wedge$" e "$\vee$" são únicos no corpo do conjunto de cláusulas e no corpo das cláusulas respectivamente, os mesmos são omitidos na notação adotada. Assim, um conjunto H de cláusulas, que representa uma conjunção destas, é notado por $\langle C_1, C_2, \ldots, C_n\rangle$ (onde fica subentendido o conectivo "$\wedge$" unindo as cláusulas), e uma cláusula C, que é uma disjunção de literais, é notada por $[L_1, L_2, \ldots, L_n]$ (onde fica subentendido conectivo "$\vee$" unindo os literais).

Algoritmos para transformação de uma fórmula qualquer para a forma normal conjuntiva (e para a forma normal disjuntiva, apresentada a seguir) são bastante usuais (e.g. [Nil82]). Estes algoritmos implicam a retirada de alguns operadores e quantificadores lógicos da fórmula para adaptá-la ao subconjunto de operadores estipulado pela forma normal conjuntiva. A retirada dos quantificadores envolve cuidados, como a troca de nomes de variáveis e a introdução de *funções de Skolem* em substituição a variáveis quantificadas existencialmente [Rob65] [RN95]. Por exemplo, uma fórmula $\alpha \rightarrow \beta$ pode ser substituída por $\neg\alpha \vee \beta$ sem prejuízo semântico, procedendo assim à eliminação do operador de implicação.

Exemplo 6 *Se voltarmos a atenção à frase proferida pela mãe de Calvin no exemplo 3, vemos que ela foi expressa diretamente em forma normal conjuntiva. A frase original "ou eu ainda estou sonhando ou ele está tocando música clássica a 78 rpm" apresenta dois literais (L_1 e L_2) unidos pelo conectivo $\vee$, sendo L_1=ela está sonhan-*

[4]Literal: uma fórmula atômica ou sua negação. Fórmula atômica: os símbolos "V" ou "F", um símbolo proposicional ou ainda, um símbolo de predicado enário tendo como parâmetros n termos.

do e L_2=Calvin está tocando música clássica a 78 rpm. Vemos também que, ao ser apresentada como premissa (i) no exemplo 3, ela foi transformada da forma normal conjuntiva original para a forma geral utilizando o operador de implicação. Lá, a premissa (i) foi apresentada na forma "Se ela não está sonhando ele (Calvin) está tocando música clássica a 78 rpm" que representa a fórmula $\neg L_1 \rightarrow L_2$. Para retorná-la para a forma normal conjuntiva, de acordo com a regra de eliminação do operador de implicação, deve-se substitui-la por $\neg(\neg L_1) \vee L_2$, que pode ser simplificada para $L_1 \vee L_2$ (que é a forma original). O conjunto de fórmulas do exemplo, se transformado para a forma normal conjuntiva, torna-se

$$\begin{array}{c} \langle \quad [Sonhando(mãe), Tocando(Calvin)], \\ [\neg Acordada(mãe), \neg Sonhando(mãe)], \\ [Acordada(mãe)] \quad \rangle \end{array}$$

onde as frases foram transformadas em fórmulas lógicas[5].

Outra forma normal encontrada na lógica, de certa forma simétrica à forma normal conjuntiva, é denominada *Forma Normal Disjuntiva* ou *Forma Clausal Dual.* Para sua definição, basta a inversão da conjunção pela disjunção e vice-versa. Assim, um conjunto de fórmulas está em forma normal disjuntiva se ele está na forma $D_1 \vee D_2 \vee \ldots \vee D_n$, onde cada D_i é uma cláusula dual. Uma fórmula é uma cláusula dual se está na forma $L_1 \wedge L_2 \wedge \ldots \wedge L_n$, onde cada L_i é um literal. Um conjunto H de cláusulas duais, representando uma disjunção destas, é notado por $[D_1, D_2, \ldots, D_n]$ (onde fica subentendido o conectivo "$\vee$" unindo as cláusulas duais), e uma cláusula dual D, que é uma conjunção de literais, é notada por $\langle L_1, L_2, \ldots, L_n \rangle$ (onde fica subentendido o conectivo "$\wedge$" unindo os literais).

As formas normais podem ser vistas como duas maneiras de expressar a mesma idéia. A forma normal conjuntiva representa as expressões lógicas da seguinte maneira: (... *ou* ... *ou* ...) *e* (... *ou* ... *ou* ...) *e* ..., e assim por diante. Por sua vez, a forma normal disjuntiva representa as mesmas expressões lógicas com outra estrutura: (... *e* ... *e* ...) *ou* (... *e* ... *e* ...) *ou* .... A forma normal disjuntiva representa as classes de modelos das premissas (representa as opções de fórmulas que têm de ser verdadeiras simultaneamente para satisfazer as premissas). Quando expressas nesta forma, apenas uma premissa tem de ser verdadeira para todo o conjunto o ser. Já,

[5] *Os símbolos Mãe e Calvin são chamados de constantes. Os símbolos Tocando, Sonhando e Acordada são chamados de predicados. Tocando(Calvin) é um literal cujo predicado é Tocando e cujo termo é a constante Calvin.*

se expressas na forma normal conjuntiva, o conjunto de premissas só é verdadeiro se todas as premissas o forem simultaneamente.

Exemplo 7 *O conjunto de premissas expresso em forma normal disjuntiva*

$$\begin{bmatrix} \langle Sonhando(mãe), \neg Acordada(mãe), Acordada(mãe)\rangle, \\ \langle Tocando(Calvin), \neg Acordada(mãe), Acordada(mãe)\rangle, \\ \langle Sonhando(mãe), \neg Sonhando(mãe), Acordada(mãe)\rangle, \\ \langle Tocando(Calvin), \neg Sonhando(mãe), Acordada(mãe)\rangle \end{bmatrix}$$

representa o mesmo conjunto de premissas expresso na forma normal conjuntiva no exemplo 6.

Uma fórmula representada em uma das formas pode ser transformada para a outra. Nota-se, pelo exemplo anterior, que para a transformação de uma fórmula de uma forma canônica para outra, apenas a distributividade dos operadores $\vee$ e $\wedge$ foi aplicada. Ambas as formas são equivalentes. Na verdade, uma fórmula lógica W, sua forma normal conjuntiva W_c e sua forma normal disjuntiva W_d são equivalentes:

$$W \Leftrightarrow W_c \Leftrightarrow W_d$$

Além de sua definição, as formas normais canônicas apresentam outras duas características simétricas:

- **Escopo das Variáveis**

  Com a eliminação dos quantificadores universais e existenciais de uma fórmula, ao ser transformada para uma das formas canônicas, fica garantido que toda variável de uma cláusula ou cláusula dual é quantificada universalmente. Porém, o escopo das variáveis não é o mesmo nas duas formas.

  Em um conjunto de cláusulas o escopo de cada variável é a própria cláusula. Assim, se uma variável x_1 por exemplo ocorre em duas cláusulas, cada ocorrência é independente, como se fossem duas variáveis diferentes (somente com o nome igual). Isto geraria um problema de ambigüidade ao ser aplicada uma substituição, por exemplo, na variável x_1. Para evitar isto, ao ser utilizada a forma normal conjuntiva, costuma-se *renomear as variáveis* de modo que um mesmo nome de variável não se repita em mais de uma cláusula.

Por sua vez, na forma normal disjuntiva o escopo das variáveis é global, ou seja, este abrange todo o conjunto de cláusulas duais. De uma certa forma, pode-se entender que, em cada cláusula dual, tem-se "um fragmento" da informação de todo o conjunto de cláusulas (é como em um holograma onde cada parte contém uma fração do todo). Ao transformar-se um conjunto de cláusulas para um conjunto de cláusulas duais, toda variável que ocorre em uma cláusula é "espalhada" pelas cláusulas duais durante o processo de transformação. Assim, se uma variável ocorre em mais de uma cláusula dual, ela é na realidade a mesma variável (pois teve origem na mesma cláusula).

- **Tautologias e Contradições**

  Uma cláusula é uma *tautologia* se ela contém literais complementares, isto é, um literal e sua negação.

  Como na forma normal conjuntiva as cláusulas estão unidas pelo conectivo "$\wedge$", para todo o conjunto de cláusulas ser verdadeiro em uma interpretação, cada cláusula tem de ser, independentemente, verdadeira nesta interpretação. Portanto, uma cláusula tautológica pode ser retirada de um conjunto de cláusulas, sem alterar o valor verdade deste.

  Já uma cláusula dual é uma *contradição* se ela contém literais complementares, isto é, um literal e sua negação.

  Como na forma normal disjuntiva as cláusulas duais estão unidas pelo conectivo "$\vee$", para o conjunto ser verdadeiro em uma interpretação, apenas uma cláusula dual tem de ser verdadeira nesta interpretação. Portanto, uma cláusula dual contraditória pode ser retirada de um conjunto de cláusulas duais, sem alterar o valor verdade deste.

Exemplo 8 *A cláusula [Gosta(Calvin,Haroldo), $\neg$Gosta(Calvin,Haroldo)] é uma tautologia. Isto pode ser comprovado de duas formas: (i) em uma cláusula os literais estão unidos pelo conectivo "$\vee$". Pelo princípio do terceiro excluído, sob qualquer interpretação a afirmação "Calvin gosta de Haroldo* **ou** *Calvin não gosta de Haroldo" é verdadeira; e (ii) a fórmula expressa pela cláusula pode ser transformada para a fórmula $\neg Gosta(Calvin, Haroldo) \rightarrow \neg Gosta(Calvin, Haroldo)$, que também é verdadeira sob qualquer interpretação.*

Simetricamente, a cláusula dual

$$\langle Gosta(Calvin, Haroldo), \neg Gosta(Calvin, Haroldo) \rangle$$

é uma contradição. Isto pode ser comprovado, também de duas formas: (i) em uma cláusula dual os literais estão unidos pelo conectivo "$\wedge$". Pelo princípio do terceiro excluído, sob nenhuma interpretação a afirmação "Calvin gosta de Haroldo **e** *Calvin não gosta de Haroldo" pode ser verdadeira; e (ii) a fórmula expressa pela cláusula dual pode ser transformada para a fórmula* $\neg(Gosta(Calvin, Haroldo) \rightarrow Gosta(Calvin, Haroldo))$, *que também não pode ser verdadeira sob nenhuma interpretação.*

Os principais métodos de prova automática de teoremas – o método da *Resolução* e o método dos *Tableaux Semânticos* – têm uma estreita relação com as formas normais canônicas da lógica. O método de resolução exige que hipóteses e teorema negado sejam transformados para a forma normal conjuntiva antes que o processo de prova seja iniciado. Já o método dos Tableaux consiste exatamente na transformação das hipóteses e do teorema negado para a forma normal disjuntiva. As formas canônicas também são utilizadas em simplificação de circuitos digitais e em sistemas de representação de conhecimento baseados em lógica [Dah83].

2.3.3 SUBSUNÇÃO

Uma característica comumente encontrada em métodos de prova automática baseados na forma normal conjuntiva é a retenção de cláusulas intermediárias. Ela é inerente ao processo de prova por refutação, que pode ser descrito da seguinte forma: baseado em um conjunto Δ de cláusulas é gerado um conjunto δ de conseqüências lógicas. Se a contradição é alcançada, então o teorema está provado. Senão δ é acrescido a Δ e o ciclo é recomeçado. Com isto, Δ cresce, no pior caso, exponencialmente. Não se deve esquecer que é fundamental a um método de prova automática a busca pela eficiência, tornando-se imprescindível minimizar este crescimento exponencial.

Para minimizar este crescimento, os métodos utilizam *estratégias* que direcionam o processo de prova. Às estratégias cabe restringir a aplicação das regras de inferência, restringindo assim a criação de novas cláusulas. Além disto, os métodos utilizam algoritmos de simplificação do conjunto de cláusulas retidas, como por exemplo a *subsunção* e a *demodulação*. A subsunção é utilizada para evitar a retenção de cláusulas cuja informação é "logicamente mais fraca" que a informação de outras cláusulas do conjunto. Se, no conjunto destas, estiver contida a relação de igualdade, a demodulação é utilizada para reescrever as cláusulas, padronizando e simplificando sua informação.

Este trabalho utiliza a subsunção. A subsunção é um tipo especial de implicação (do inglês "entailment") entre cláusulas, facilmente reconhecível sintaticamente. Uma

cláusula C_1 subsume uma cláusula C_2, se C_2 for uma instância, com ou sem literais acrescidos, de C_1. Como a cláusula subsumida é uma conseqüência lógica da que a subsume, sempre que esta for válida aquela também será. Deste modo, é possível retirar a cláusula subsumida do conjunto original sem alterar a satisfazibilidade deste. Cabe ressaltar que duas cláusulas idênticas ou *variantes alfabéticas* (isto é, cláusulas que se diferenciam apenas pelos nomes de suas variáveis) subsumem-se mutuamente.

Formalmente, uma cláusula C_1 *subsume* uma cláusula C_2 se existe uma substituição θ tal que[6]:

$$C_1\theta \subset C_2$$

Como a subsunção é um assunto para o qual resolveu-se incluir um grande número de exemplos, estes tiveram de ser reduzidos para exemplificar apenas o fato abordado em cada item. Por isto optou-se por abandonar momentaneamente os exemplos utilizados até agora, e apresentar outros exemplos menores, na forma de cláusulas e literais genéricos.

Exemplo 9 *A cláusula* $[P(x_1), R(a)]$ *subsume a cláusula* $[P(a), R(a), Q(a)]$ *com* $\theta = \{(x_1/a)\}$. *No entanto a cláusula* $[P(x_1), R(a)]$ *não subsume a cláusula* $[P(a), R(x_2)]$, *pois nenhuma substituição aplicada àquela pode torná-la igual a esta.*

Esta definição de subsunção pode ser estendida para cláusulas duais: uma cláusula dual D_1 *subsume* uma cláusula dual D_2 se existe uma substituição θ tal que:

$$D_1 \dot{\subset} D_2\theta$$

Devido ao escopo das variáveis em um conjunto de cláusulas duais ser global, isto é, todo o conjunto, assumimos que cláusulas duais que são variantes alfabéticas não se subsumem.

Exemplo 10 *A cláusula dual* $\langle P(a), R(a)\rangle$ *subsume a cláusula dual* $\langle P(x_1), R(x_2), Q(a)\rangle$, *com* $\theta = \{(x_1/a), (x_2/a)\}$.

O mesmo conceito pode ser usado para a definição de subsunção entre literais de uma cláusula. Dada uma cláusula $C = [L_1, L_2, ..., L_n]$, e dois literais $L_i, L_j \in C$ que não são variantes alfabéticas, L_i subsume L_j se existir uma substituição θ tal que $L_i = L_j\theta$. Ressaltamos que literais idênticos subsumem-se entre si. Como o escopo das variáveis que ocorrem na cláusula C restringe-se a ela, se L_i e L_j forem variantes alfabéticas

[6] A notação Pθ significa a aplicação da substituição θ na fórmula P.

e suas variáveis não ocorrerem em outros literais de C, então L_i e L_j subsumem-se mutuamente, já que suas variáveis poderiam ser simplesmente renomeadas e tornadas iguais sem afetar os demais literais da cláusula.

Exemplo 11 *A cláusula* $[P(x_1), P(a)]$ *é equivalente à cláusula* $[P(a)]$, *pois o literal* $P(a)$ *subsume o literal* $P(x_1)$ *com* $\theta = \{(x_1/a)\}$.

Estendemos mais uma vez o conceito para a definição de subsunção de literais de uma cláusula dual. Dada uma cláusula $D = \langle L_1, L_2, ..., L_n \rangle$, e dois literais $L_i, L_j \in D$ que não são variantes alfabéticas, L_i subsume L_j se existir uma substituição θ tal que $L_i\theta = L_j$. Também aqui literais idênticos subsumem-se entre si.

Exemplo 12 *A cláusula* $[P(x_1), P(a)]$ *é equivalente à cláusula* $[P(x_1)]$, *pois o literal* $P(x_1)$ *subsume o literal* $P(a)$ *com* $\theta = \{(x_1/a)\}$.

Uma cláusula (ou cláusula dual) é *pura* se não apresenta subsunção entre seus literais. A deleção de subsunções de um conjunto de cláusulas leva à obtenção de um conjunto com apenas cláusulas puras não subsumidas entre si, o que é chamado na literatura de *implicantes primários* (do inglês "prime implicants") do conjunto original.

2.4 O MÉTODO POR TRANSFORMAÇÃO DUAL

Antes de apresentar o método proposto, vamos analisar dois conjuntos de premissas apresentados em exemplos anteriores:

- **Caso 1**

  Pelas duas formas clausais das premissas apresentadas nos exemplos 6 e 7:

$$\begin{aligned} W_c \;=\; \langle\, & [Sonhando(\text{mãe}), Tocando(Calvin)], \\ & [\neg Acordada(\text{mãe}), \neg Sonhando(\text{mãe})], \\ & [Acordada(\text{mãe})] \,\rangle \end{aligned}$$

$$\begin{aligned} W_d \;=\; [\; & \langle Sonhando(\text{mãe}), \neg Acordada(\text{mãe}), Acordada(\text{mãe})\rangle, \\ & \langle Tocando(Calvin), \neg Acordada(\text{mãe}), Acordada(\text{mãe})\rangle, \\ & \langle Sonhando(\text{mãe}), \neg Sonhando(\text{mãe}), Acordada(\text{mãe})\rangle, \\ & \langle Tocando(Calvin), \neg Sonhando(\text{mãe}), Acordada(\text{mãe})\rangle \;] \end{aligned}$$

percebe-se que as três primeiras cláusulas duais de W_d são explicitamente contraditórias. Nenhuma interpretação pode satisfazer estas cláusulas duais. Para W_d ser satisfazível, a sua última cláusula dual tem de sê-lo. Ao serem retiradas as cláusulas duais contraditórias, tem-se:

$$W'_d = [\ \langle Tocando(Calvin), \neg Sonhando(\text{mãe}), Acordada(\text{mãe})\rangle\ ]$$

cuja forma normal conjuntiva

$$W'_c = \langle\ [Tocando(Calvin)]\ ,\ [\neg Sonhando(\text{mãe})]\ ,\ [Acordada(\text{mãe})]\ \rangle$$

contém os mesmos teoremas obtidos na prova do exemplo 3 exibida na figura 2.2. Outras características podem ser observadas:

1. W'_c contém uma premissa que já fazia parte do conjunto original W_c. De fato, pode-se entender que W'_c não é simplesmente um conjunto de conseqüências lógicas de W_c, mas sim uma instância (ou um estado) de W_c que, a princípio, pode ser satisfazível (se W_c o for).
2. Ao retirar uma cláusula dual de seu conjunto, está-se retirando uma classe de modelos possíveis. Esta classe de modelos é definida (especificada) por literais representantes de cada cláusula do conjunto original. Com isto, está-se retirando do conjunto de classes de modelos possíveis uma das combinações de literais das cláusulas do conjunto original. Ao serem eliminadas as três cláusulas duais contraditórias, vê-se que os literais *Sonhando(mãe)* e $\neg$*Acordada(mãe)* não estão mais representados em nenhuma classe de modelos. Se fossem retirados estes literais das cláusulas de W_c, obter-se-ia um conjunto igual a W'_c.
3. A conclusão "intermediária" $\neg$*Sonhando(mãe)* foi obtida, em um único ciclo de inferência, junto com a conclusão *Tocando(Calvin)*. É como se o silogismo hipotético *Acordada(mãe)* $\rightarrow$ *Tocando(Calvin)*, implícito em W_c, tivesse também sido aplicado neste mesmo ciclo. O mesmo ocorre com "cadeias" de silogismos hipotéticos. Por exemplo, se aplicado o mesmo processo de transformação e retirada de cláusulas duais contraditórias no conjunto de fórmulas P, (P $\rightarrow$ Q), (Q $\rightarrow$ R), (R $\rightarrow$ S), têm-se geradas, em um único ciclo de inferência, a conclusão (Q) obtida por modus ponens e as conclusões (R) e (S) obtidas pelas "cadeias" de silogismo hipotético.

- **Caso 2**

Analisando-se as duas formas clausais das premissas apresentadas no exemplo 5:

$$W_c = \langle\, [\neg Gosta(x,y), Amigo(x,y)]\,,\, [Gosta(Calvin, Haroldo)],\\ [Gosta(Calvin, Susie)]\,,\, [\neg Amigo(Calvin, z)]\,\rangle$$

$$W_d = [\, \langle \neg Amigo(Calvin,z), \neg Gosta(x,y),\\ Gosta(Calvin, Haroldo), Gosta(Calvin, Susie)\rangle,\\ \langle \neg Amigo(Calvin,z), Amigo(x,y),\\ Gosta(Calvin, Haroldo), Gosta(Calvin, Susie)\rangle\,]$$

percebe-se que todas as cláusulas duais são "potencialmente" contraditórias. Para torná-las contradições explícitas deve-se, nelas, aplicar uma substituição. Por exemplo, a substituição {(x/Calvin), (y/Haroldo)} torna a primeira cláusula dual contraditória. É certo que para estes valores das variáveis nenhuma interpretação pode satisfazer esta cláusula dual. Esta pode ser retirada de W_d contanto que as cláusulas restantes tenham suas variáveis substituídas pelo valor que a tornou contraditória (se as variáveis da substituição ocorrerem nas cláusulas duais restantes), pois foi com estes valores de variáveis que a contradição tornou-se explícita. Com isto, especializa-se a instância de W_c que pode ser satisfazível.

Se aplicada esta substituição em W_d e retirada a cláusula explicitamente contraditória, tem-se:

$$W'_d = [\, \langle \neg Amigo(Calvin,z), Amigo(Calvin, Haroldo),\\ Gosta(Calvin, Haroldo), Gosta(Calvin, Susie)\rangle\,]$$

Também, com a substituição {(x/Calvin), (y/Susie)}, a primeira cláusula dual de W_d pode ser tornada explicitamente contraditória. Se aplicada esta substituição em W_d e retirada a cláusula explicitamente contraditória, tem-se:

$$W''_d = [\ \langle \neg Amigo(Calvin, z), Amigo(Calvin, Susie), Gosta(Calvin, Haroldo), Gosta(Calvin, Susie) \rangle\ ]$$

Já a segunda cláusula dual de W_d é tornada explicitamente contraditória pela substituição {(x/Calvin), (z/y)}. Se aplicada esta substituição em W_d e retirada a cláusula explicitamente contraditória, tem-se:

$$W'''_d = [\ \langle \neg Amigo(Calvin, y), \neg Gosta(Calvin, y), Gosta(Calvin, Haroldo), Gosta(Calvin, Susie) \rangle,$$

Cada novo W_d dá origem a uma instância especializada de W_c. Assim, W'_d dá origem a:

$$W'_c = \langle\ [Amigo(Calvin, Haroldo)]\ ,\ [Gosta(Calvin, Haroldo)],\ [Gosta(Calvin, Susie)]\ ,\ [\neg Amigo(Calvin, z)]\ \rangle$$

Todas estas substituições podem ser combinadas e, com isto, tem-se que as duas cláusulas duais podem tornar-se contraditórias, simultaneamente, com duas substituições diferentes: {(z/y), (x/Calvin), (y/Haroldo)} e {(z/y), (x/Calvin), (y/Susie)}. Isto indica que, para estes valores de variáveis, todo W_d é explicitamente contraditório. Como na forma normal conjuntiva toda variável é quantificada universalmente, pode-se dizer que W_d é contraditório (insatisfazível). Aplicando-se estas duas substituições em W_d e eliminando as cláusulas duais explicitamente contraditórias, W_d tornar-se-ia vazio. De fato, se fosse efetuada a transformação de um conjunto vazio de cláusulas duais para a forma normal conjuntiva, seria gerado um conjunto contendo uma única cláusula vazia. Esta cláusula vazia só pode ser conseqüência lógica de um conjunto de cláusulas insatisfazível [Fit90]. Se aplicadas ao teorema [¬ Amigo(Calvin,z)], cada uma destas duas substituições, produzem as "respostas" à questão expressa pelo mesmo: "Quem é amigo de Calvin?": Haroldo e Susie.

Com estes dois casos analisados pode-se dizer que uma regra de inferência foi definida: pela forma normal disjuntiva de um conjunto de fórmulas pode-se inferir um

novo conjunto de cláusulas (especializadas), que é uma conseqüência lógica do conjunto original de fórmulas. É interessante notar que a regra de inferência opera apenas com propriedades internas ao sistema lógico subjacente (e por isto corretas). No método proposto não existe regra de inferência externa ao sistema lógico: o método resume-se em explorar a dualidade das possíveis representações e suas semânticas.

Graças a esta característica, o método de inferência proposto é denominado *Método de Inferência baseado em Transformação Dual.* Para apresentá-lo define-se:

- $\lfloor W \rfloor$ o conjunto de implicantes primários do conjunto de cláusulas (duais) W. Para este trabalho é assumido que o cálculo dos implicantes primários de um conjunto de cláusulas compreende a retirada das cláusulas tautológicas. O mesmo cálculo para um conjunto de cláusulas duais compreende a retirada de cláusulas duais explicitamente contraditórias.

- **Dual** uma operação de transformação de um conjunto de cláusulas W_c para a sua forma normal disjuntiva W_d e vice-versa, tal que $W_d = \lfloor Dual(\lfloor W_c \rfloor) \rfloor$ e $W_c = \lfloor Dual(\lfloor W_d \rfloor) \rfloor$

- **Teoria** o par composto pelos implicantes primários das formas normais conjuntiva e disjuntiva de um conjunto de fórmulas.

- $W_c(T)$ e $W_d(T)$ os conjunto de cláusulas e cláusulas duais da teoria T, respectivamente.

- **Box** uma operação de união de teorias, tal que sendo $T_1, T_2, \ldots, T_n$ teorias e $T_b = Box(T_1, T_2, \ldots, T_n)$, tem-se $W_c(T_b) = \lfloor W_c(T_1) \cup W_c(T_2) \cup \ldots \cup W_c(Tn) \rfloor$ e $W_d(T_b) = \lfloor Dual(W_c(T_b)) \rfloor$.

- $\Theta(T)$ o conjunto de todas θ_{ij}, onde θ_{ij} é uma substituição cuja aplicação representa a i-ésima maneira de tornar a j-ésima cláusula dual da teoria T uma contradição explícita.

- $\Omega(T)$ o conjunto de todas ω_i, onde ω_i é a i-ésima substituição que torna uma ou mais cláusulas duais da teoria T uma contradição explícita, ou seja, Ω é o conjunto de todas as substituições de Θ mais as resultantes de combinações compatíveis[7] entre elas.

[7]Uma substituição é um mapeamento do conjunto de variáveis para o conjunto de termos. Uma substitui'cão é válida se cada uma de suas variáveis é mapeada para um, e apenas um, termo e se o mapeamento de suas variáveis não apresenta ciclos. Duas substituições são compatíveis se a substituição resultante da composição entre as duas é válida.

- **Inference(T,ω_i)** uma função que aplica a regra de inferência utilizada pelo método na teoria T. Seu retorno é uma nova teoria T_i que representa o "estado" inferido pela aplicação de ω_i em $W_d(T)$ e posterior retirada das cláusulas duais explicitamente contraditórias, ou seja, $W_d(T_i) = \lfloor W_d(T)\omega_i \rfloor$ e $W_c(T_i) = \lfloor Dual(W_d(T_i)) \rfloor$.

Baseado nestas definições, o método de inferência proposto é apresentado na definição da rotina *Pulsar(Teoria)*:

Pulsar(T)
renomear variaveis
se $\Omega(T) = \emptyset$
  então: **retornar "teoria fechada" (*)**
  senão: $\Delta \leftarrow \{T\}$
    para cada ω_i de $\Omega(T)$:
      $\delta \leftarrow$ Inference(T,ω_i)
      se $W_d(\delta) = \emptyset$
        então: **retornar "teoria contraditoria" (**)**
        senão: $\Delta \leftarrow \Delta \cup \delta$
    $T_{box} \leftarrow Box(\Delta)$
    se $T = T_{box}$
      então: **retornar "teoria fechada" (***)**
      senão: Pulsar(T_{box})

Observa-se que a execução da rotina Pulsar apresenta três possibilidades:

1. a teoria T é fechada e após um número finito de ciclos (chamadas recursivas) a execução é encerrada porque: ou nenhuma cláusula dual de T é contraditória (*) ou as novas teorias geradas (T_{box}) não acrescentam nenhuma cláusula nova a T (***).

2. a teoria T é contraditória e após um número finito de ciclos é gerada uma substituição que torna todas as cláusulas duais de T explicitamente contraditórias, causando o final da execução (**).

3. a execução nunca termina devido à semi-decidibilidade da lógica de primeira ordem.

A rotina Pulsar descreve um método de inferência correto e completo (para refutação). Ela pode também ser utilizada como um método de prova dedutivo (um gerador de teoremas), conforme mostra o exemplo 13:

Exemplo 13 *A figura 2.5 mostra uma execução da rotina Pulsar para uma teoria não contraditória. A teoria T inicial é a mesma analisada no caso 1:*

$$W_c = \langle\, [Sonhando(mãe), Tocando(Calvin)]\,,\, [\neg Acordada(mãe), \neg Sonhando(mãe)],\\ [Acordada(mãe)]\,\rangle$$

$$W_d = [\, \langle Sonhando(mãe), \neg Acordada(mãe), Acordada(mãe)\rangle,\\ \langle Tocando(Calvin), \neg Acordada(mãe), Acordada(mãe)\rangle,\\ \langle Sonhando(mãe), \neg Sonhando(mãe), Acordada(mãe)\rangle,\\ \langle Tocando(Calvin), \neg Sonhando(mãe), Acordada(mãe)\rangle\,]$$

Como suas claúsulas são todas fechadas, as contradições são explícitas. Então, a única substituição utilizada para a inferência é a substituição vazia ($\{\}$) que resulta na teoria T1:

$$W_c = \langle\, [Tocando(Calvin)]\,,\, [\neg Sonhando(mãe)]\,,\, [Acordada(mãe)]\,\rangle$$

$$W_d = [\, \langle Tocando(Calvin), \neg Sonhando(mãe), Acordada(mãe)\rangle\,]$$

Para terminar este ciclo de inferência, as teorias T e T1 são "unidas" através da operação Box que resulta na teoria T_{box}. Como as cláusulas de T1 subsumem todas as cláusulas de T, T_{box} é igual a T1. O ciclo é reiniciado através da chamada recursiva da rotina Pulsar(T_{box}). Como esta teoria não possui nenhuma cláusula dual contraditória, é concluído que a teoria é fechada e a execução da rotina é encerrada.

A rotina Pulsar também pode ser utilizada como um método de prova por refutação, confome é mostrado no exemplo 14:

Exemplo 14 *Vejamos como ficaria a execução da rotina Pulsar para uma teoria contraditória. A teoria T inicial é a mesma analisada no caso 2:*

$$W_c = \langle\, [\neg Gosta(x,y), Amigo(x,y)]\,,\, [Gosta(Calvin, Haroldo)],\\ [Gosta(Calvin, Susie)]\,,\, [\neg Amigo(Calvin, z)]\,\rangle$$

$$W_d = [\, \langle \neg Amigo(Calvin, z), \neg Gosta(x,y),\\ Gosta(Calvin, Haroldo), Gosta(Calvin, Susie)\rangle,\\ \langle \neg Amigo(Calvin, z), Amigo(x,y),\\ Gosta(Calvin, Haroldo), Gosta(Calvin, Susie)\rangle\,]$$

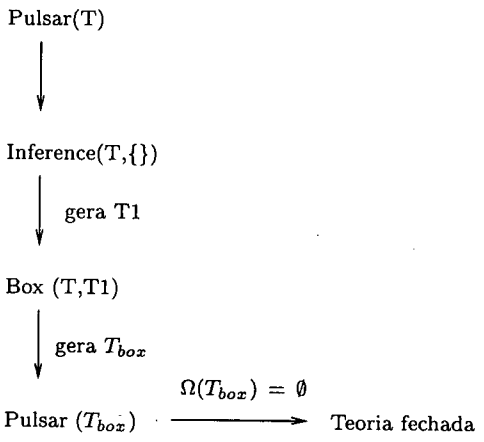

Figura 2.5: Execução da rotina Pulsar para o exemplo 13

viu-se que as seguintes substituições tornam cláusulas duais desta teoria explicitamente contraditórias:

- *as substituições {(x/Calvin), (y/Haroldo)} e {(x/Calvin), (y/Susie)} tornam a primeira cláusula dual contraditória.*
- *a substituição {(x/Calvin), (z/y)} torna a segunda cláusula dual contraditória.*
- *as substituições {(z/y), (x/Calvin), (y/Haroldo)} e {(z/y), (x/Calvin), (y/Susie)}, resultantes da combinação das anteriores, tornam as duas cláusulas duais contraditórias simultaneamente.*

Como no conjunto $\Omega(T)$ existem substituições que tornam todo o conjunto de cláusulas duais de T contraditório simultaneamente, e o objetivo é refutar a teoria T, as demais substituições podem ser desconsideradas. Assim, pela inferência, a teoria com W_d vazio é gerada, causando o final da execução da rotina. Para saber quem são os amigos de Calvin basta aplicar as substituições que tornam todo $W_d(T)$ contraditório na variável do teorema e tem-se `z = Haroldo` *e* `z = Susie`.

2.5 CONCLUSÃO

Abordou-se o aspecto dedutivo da lógica e as características gerais dos métodos lógicos de prova de teoremas. Apresentou-se noções lógicas comumente exploradas em métodos automáticos de prova de teoremas como as formas normais canônicas das fórmulas lógicas e a subsunção entre cláusulas.

Dados os fundamentos, o *Método de Inferência baseado em Transformação Dual* foi apresentado. Este método foi assim denominado devido à sua inferência dar-se exclusivamente pela exploração das propriedades das duas representações canônicas de uma fórmula, sem utilizar nenhuma regra de inferência externa ao sistema lógico subjacente. Uma outra característica do método é a diminuição dos ciclos de inferência necessários à obtenção de uma prova.

Apesar destas características, o método mostra-se bastante ineficiente para implementação. O mesmo apresenta dois passos que contribuem diretamente para tal: a transformação entre as formas canônicas (representada pela operação *Dual* da rotina Pulsar) e a união das teorias geradas (representada pela operação *Box* da mesma rotina).

A fim de melhorar a eficiência do método foram propostos em [Bit97a] e [Bit96a] algoritmos que implementam as operações *Dual* e *Box* respectivamente. Destes, o algoritmo que implementa a operação *Dual* é apresentada no capítulo subseqüente. Já o algoritmo que implementa a operação de *Box* não é abordado neste trabalho.

Capítulo 3

Um Algoritmo para Transformação Dual

No capítulo anterior foi citado como um dos passos "caros" do método de inferência proposto, a *transformação dual*, isto é, a transformação entre as formas normais canônicas conjuntiva e disjuntiva de fórmulas da lógica de primeira ordem.

Neste capítulo é abordado o problema desta transformação e é apresentado um algoritmo que é capaz de realizar esta transformação de forma eficiente.

3.1 A Transformação Dual

As formas normais canônicas conjuntiva (W_c) e disjuntiva (W_d) apresentam uma relação estrutural entre elas, sendo uma obtida pela distributividade dos operadores da outra. Com isto, cada cláusula da forma normal conjuntiva é formada por um (e apenas um) representante de cada cláusula dual da forma normal disjuntiva, e vice-versa. Com isto, também, todas as cláusulas de uma forma normal encontram-se representadas simultaneamente em cada cláusula da outra forma (como mostra o exemplo 15). Cada uma das formas normais é um tipo de representação holográfica da outra: cada "parte" de uma contém uma representação do "todo" da outra.

A maneira mais direta de obter W_d dado W_c é construir todas as combinações possíveis de literais de W_c, de modo que em cada combinação apareça apenas um literal de cada cláusula de W_c. O problema com este método é que o número de cláusulas duais resultantes cresce exponencialmente com o número de cláusulas de W_c e, daquelas, muitas são subsumidas. O conjunto de cláusulas duais W_d deve então ser *simplificado* ou seja, devem ser calculados os seus implicantes primários (como mostra o exemplo 16).

O caso dual[1] é simétrico. Seja qual for a forma canônica do conjunto original, aplicando-se a transformação dual em um conjunto resultante de uma transformação dual, obtém-se, após a devida simplificação, o conjunto original.

Exemplo 15 *A transformação do conjunto de cláusulas:*

$$W_c = \langle [A1, A2, A3], [B1, B2, B3] \rangle$$

resulta no conjunto de cláusulas duais:

$$W_d = [\ \langle A1, B1\rangle, \langle A1, B2\rangle, \langle A1, B3\rangle, \langle A2, B1\rangle, \langle A2, B2\rangle \\ \langle A2, B3\rangle, \langle A3, B1\rangle, \langle A3, B2\rangle, \langle A3, B3\rangle\ ]$$

Cada cláusula dual contém todas as cláusulas representadas, por isto, o número de literais de cada cláusula dual é o mesmo que o número de cláusulas de W_c.
Já, na transformação inversa, 512 cláusulas diferentes são geradas. Porém muitas destas não são puras ou são subsumidas. Por exemplo, dentre as 512 cláusulas geradas tem-se [A1, A1, A1, A2, A2, A2, A3, A3, A3], [A1, A1, A2, A2, A2, A3, A3, A3, B1] e [A1, A2, A3, B1, B1, B2, B2, B2, B3] com um literal oriundo de cada cláusula dual. Destas, a primeira pode ser simplificada para [A1, A2, A3] que subsume a segunda e terceira cláusulas.
Se calculado o conjunto de implicantes primários deste conjunto de 512 cláusulas, obtém-se um conjunto igual a W_c. *Apesar das cláusulas de* W_c *não conterem nove literais cada, todas as nove cláusulas duais de* W_d *está representada em cada cláusula de* W_c. *Ao torná-las puras, um mesmo literal destas ou representa mais de uma cláusula dual de* W_d *ou subsume um literal de uma claúsula dual deste.* **Esta noção de literais representantes de cláusulas bem como a noção de que toda cláusula está representada (mesmo que não aparentemente) em cada cláusula de seu estado dual, é muito importante para o entendimento do algoritmo proposto.**

Exemplo 16 *A transformação do conjunto de cláusulas:*

$$W_c = \langle [P(a), Q(a)], [P(a), R(x_1)], [P(x_2), R(a)] \rangle$$

para a sua forma normal disjuntiva, numa primeira etapa gera o seguinte conjunto de cláusulas duais:

$$W_d = [\ \langle P(a), P(a), P(x_2)\rangle, \quad \langle P(a), P(a), R(a)\rangle, \quad \langle P(a), R(x_1), P(x_2)\rangle$$

[1] Neste trabalho o termo *dual* é utilizado para indicar a transformação inversa entre formas canônicas ou a forma canônica oposta em si. Assim, por exemplo, o *dual* da transformação de W_c para W_d é a transformação de W_d para W_c; e o *dual* de W_c é W_d.

$$\langle P(a), R(x_1), R(a)\rangle, \quad \langle Q(a), P(a), P(x_2)\rangle, \quad \langle Q(a), P(a), R(a)\rangle$$
$$\langle Q(a), R(x_1), P(x_2)\rangle, \quad \langle Q(a), R(x_1), R(a)\rangle \ ]$$

que com a simplificação torna-se:

$$W_d = [\ \langle P(x_2)\rangle,\ \langle P(a), R(a)\rangle,\ \langle Q(a), R(x_1)\rangle\ ]$$

A transformação dual, independentemente de sua aplicação na rotina *Pulsar* (definida no capítulo anterior), representa um importante objeto de pesquisa (por exemplo, [Soc91]). Algoritmos para transformação de uma fórmula qualquer para a forma clausal são bastante usuais (por exemplo [Nil82]), já um algoritmo para a transformação entre as duas formas clausais foi proposto originalmente por Quine [Qui59]. Mais recentemente, um algoritmo mais eficiente foi proposto [Soc91]. No entanto, este algoritmo gera algumas cláusulas subsumidas, o que torna obrigatória a verificação de subsunção para cada nova cláusula gerada.

3.2 O ALGORITMO DE TRANSFORMAÇÃO DUAL

Na seção entitulada *Current State of the art: Basic research problems to solve* de [WV94], o problema da verificação da subsunção entre cláusulas é colocado, pelo autor, como um dos dois mais importantes problemas a serem resolvidos em Prova Automática de Teoremas. O problema é colocado da seguinte forma: por um lado, o tempo de processamento requerido para a verificação da subsunção representa um grande percentual do tempo total de processamento necessário à solução de um problema. Por outro lado, se relaxada esta verificação, o espaço (memória) necessário ao armazenamento de cláusulas subsumidas representa um grande percentual do espaço total necessário à solução de um problema.

Em [Bit97a] o autor apresenta outro problema mais genérico: "(...) Uma dificuldade inerente a qualquer método de prova automático de teoremas é a conhecida por *Explosão Combinatória*, isto é, o espaço e tempo necessários à solução do problema cresce exponencialmente com o tamanho do mesmo. Uma solução possível para esta dificuldade é o uso de computadores parelelos [Qui87]. O uso de computadores paralelos implica que algoritmos devem ser redesenhados a fim de explorar eficientemente a capacidade de paralelismo da máquina. No caso de provadores automáticos de teoremas, métodos como *Graph Resolution* [Eis86] foram adaptados com sucesso para computadores paralelos [LM86] mas, normalmente, provadores de teoremas não são fáceis de serem paralelizados. Uma solução alternativa é projetar algoritmos específicos

para arquiteturas paralelas (por exemplo [CF91] e [MV88]).”

O algoritmo proposto vem ao encontro da minimização destes dois problemas:

- ele gera, a partir de um conjunto de cláusulas (duais), diretamente o conjunto de implicantes primários de seu conjunto dual, sem gerar cláusulas (duais) subsumidas e,
- sua estrutura o torna naturalmente concorrente e, por isto, facilmente paralelizável.

Nele são levadas em conta as informações sobre o contexto de cada cláusula (dual) e de cada literal do conjunto original. Estas informações são armazenadas em uma estrutura de dados na forma de um *hipercubo*[2] onde dá-se a *propagação* das informações que formarão as novas cláusulas (duais). Pode-se dizer que o algoritmo trabalha com as características “geométricas” de um conjunto de cláusulas.

As explicações e exemplos são apresentados para a transformação de um conjunto de cláusulas para o conjunto de cláusulas duais equivalente. Isto foi adotado para evitar a constante repetição de notas indicando que o caso dual é simétrico. Esclarece-se então, de antemão, que o algoritmo é totalmente simétrico para os dois casos de transformação.

3.2.1 DINÂMICA DO ALGORITMO

Antes de falar da dinâmica do algoritmo, é necessário definir a estrutura de dados básica na qual se dá esta dinâmica. Esta estrutura de dados é definida na forma de um hipercubo de dimensão n onde n é igual ao número de predicados do conjunto original de cláusulas. Cada eixo do hipercubo representa um destes predicados; assim, cada vértice de dimensão n representa o cruzamento de n predicados. A cada vértice é dado uma *tarja* que é um vetor de tamanho n onde cada posição representa um eixo do hipercubo. Uma posição i deste vetor é igual a 1 se o eixo i compõe o vértice da tarja. Assim, a tarja de um vértice de dimensão n contém n posições iguais a 1.

Exemplo 17 *O hipercubo associado aos exemplos 6 e 7 é mostrado na figura 3.1. Neste exemplo o conjunto de cláusulas é composto por 3 predicados, sendo associado pois, um hipercubo de 3 dimensões (n=3). A primeira dimensão (os eixos) representa cada*

[2]Um hipercubo de dimensão n é um grafo onde os vértices são todos os elementos do conjunto de n-tuplas $(k_1, ..., k_n)$ onde $k_i \in \{0, 1\}$ e onde quaisquer dois vértices cuja representação difere em apenas uma posição são conectados entre si [Qui87].

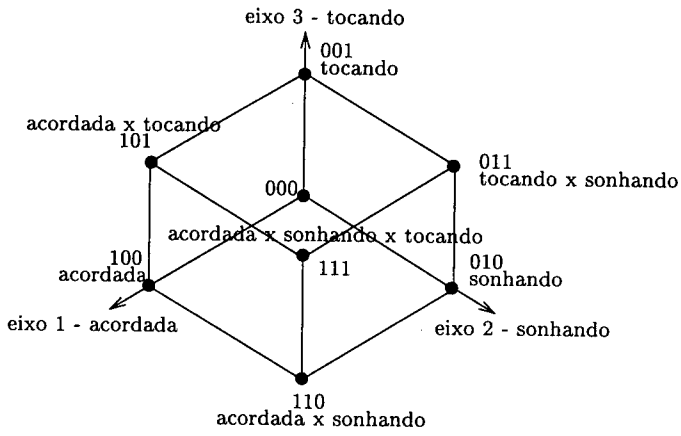

Figura 3.1: Predicados dos exemplos 6 e 7 no hipercubo

predicado isoladamente. Assim, o eixo 1 representa o predicado Acordada e o vértice de primeira dimensão composto por este eixo possui a tarja 100 já que representa apenas o eixo 1. O mesmo acontece com os demais vértices de primeira dimensão cujas tarjas são 010 e 001 por representarem os eixos 2 (com o predicado Sonhando) e 3 (com o predicado Tocando) respectivamente. Já os vértices de segunda dimensão - 101, 011 e 110 - representam o cruzamento de dois eixos ou seja, de dois predicados. No vértice final, de terceira dimensão, a tarja é 111 e representa o cruzamento dos três eixos do hipercubo (todos os predicados).

Definido o hipercubo pode-se associar cada cláusula de um conjunto a um de seus vértices. Uma cláusula pertence ao vértice que representa o cruzamento dos eixos que representam os literais que compõem esta cláusula. As cláusulas duais do exemplo 7 em seus vértices associados são mostrados na figura 3.2.

Este conjunto de cláusulas duais representa o resultado esperado pela transformação dual do conjunto de cláusulas do exemplo 6. Já as cláusulas duais em seu vértices (como mostrado na figura 3.2) representa o resultado final da dinâmica do algoritmo proposto para esta transformação dual.

Conhece-se então, o resultado final do algoritmo: cada cláusula dual em seu respectivo vértice. Na realidade, pelo algoritmo, elas são geradas em seus vértices. Vejamos como isto se dá para o conjunto de cláusulas do exemplo acima.

Assim como as cláusulas, os literais de um conjunto de cláusulas também pertencem a um vértice (sempre de primeira dimensão para o caso de literais). Cada um destes literais pertence ao vértice cujo eixo representa o predicado do literal. Para iniciar a dinâmica do algoritmo, cada vértice de primeira dimensão recebe os literais do conjunto de cláusulas relativos ao predicado representado pelo eixo do vértice (mostrado na figura

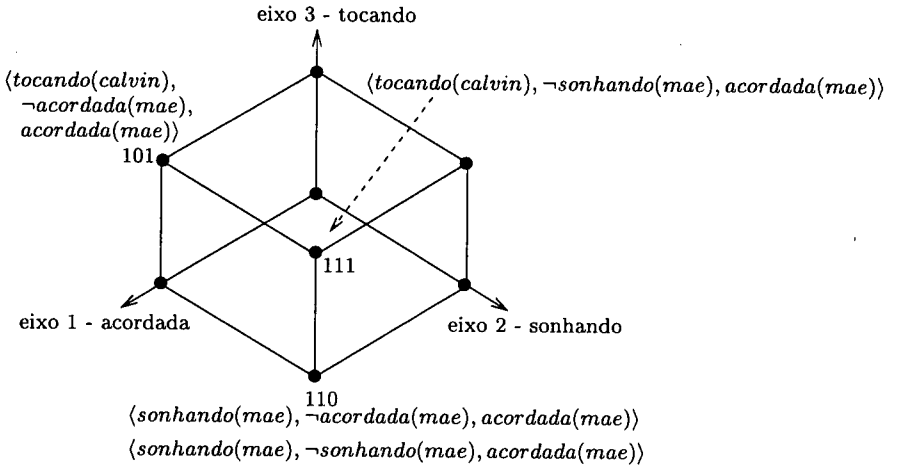

Figura 3.2: Cláusulas duais do exemplo 7 em seus vértices associados

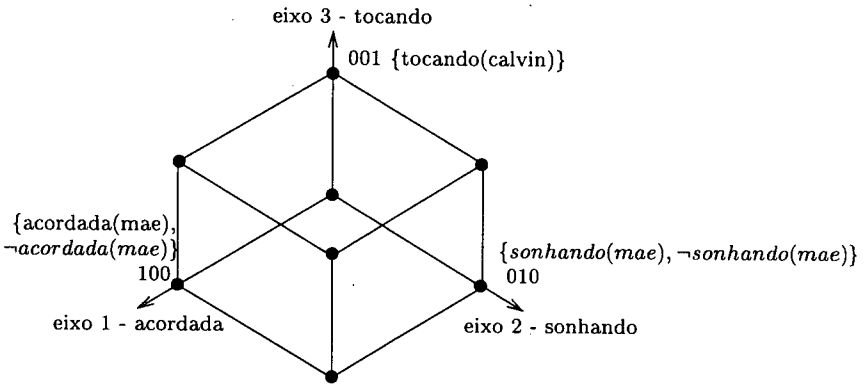

Figura 3.3: Literais das cláusulas do exemplo 7 em seus vértices associados

3.3).

Em cada um destes vértices, os literais são combinados entre eles de forma que somente sejam combinados literais que pertençam a cláusulas diferentes[3]. É então verificado se alguma claúsula dual é resultante desta combinação. Como não é o caso, estas combinacões de literais de mesmo predicado são *propagadas* para os vértices sucessores (de próxima dimensão).

Nestes vértices, as combinações de literais recebidas de vértices predecessores são combinadas resultando em novas combinações, desta vez, de literais de dois predicados diferentes (sempre evitando que literais oriundos da mesma cláusula se misturem).

[3]Observa-se na figura 3.4, por exemplo, que nenhuma combinação entre os literais *Sonhando(mãe)* e *Tocando(Calvin)* foi realizada no vértice 011, porque eles são oriundos da mesma cláusula do conjunto original.

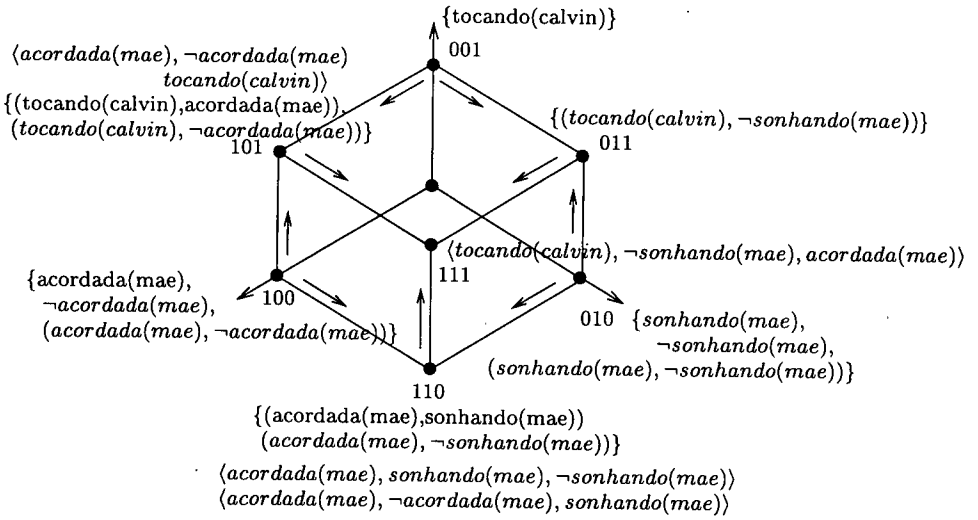

Figura 3.4: Vértices com as combinações de literais e cláusulas duais

Novamente é verificado se alguma cláusula dual foi gerada. Neste caso, nos vértices de tarja 101 e 110 foram geradas cláusulas duais. As combinações que não geraram cláusulas duais são propagadas novamente para os vértices sucessores, neste caso, o vértice 111 de terceira dimensão.

Este vértice (111) é o vértice final da propagação e recebe as combinações de todos os vértices predecessores (de segunda dimensão). Estas combinações são combinadas entre si e é, então, verificado se alguma cláusula dual foi gerada nesta combinação. No caso, apenas uma cláusula dual.

Na figura 3.4 estão mostrados em cada vértice os conjuntos das combinações propagadas para os vértices sucessores e as cláusulas duais geradas. As setas ao longo dos eixos do hipercubo indicam os caminhos da propagação.

A dinâmica do algoritmo pode ser assim resumida: para toda dimensão de 1 a n, a cada dimensão m, a cada vértice desta dimensão, literais de m símbolos de predicado são combinados segundo algumas regras e propagados para a próxima dimensão.

Em cada vértice de dimensão maior que um, a combinação é feita entre as informações recebidas de vértices predecessores. Em uma primeira vista, isto pode parecer implicar que, para realizar tal combinação, estes vértices necessitam das informações de todos os vértices predecessores. No entanto, para realizar a combinação de literais, cada um destes vértices necessita das informações de apenas dois de seus vértices predecessores.

Esta propriedade pode ser assim explicada: a informação propagada por um vértice de dimensão n, representa o cruzamento da informação de n eixos, por conseguinte, o

cruzamento de literais de n símbolos de predicados diferentes. Quaisquer dois vértices de mesma dimensão contém pelo menos um eixo diferente envolvido neste cruzamento. Também dois vértices quaiquer V_1 e V_2, sendo V_1 um dos predecessores imediatos de V_2, V_2 representa o cruzamento de um (e apenas um) eixo a mais dos já representados por V_1[4]. Então, a despeito do número de predecessores, todos os eixos cruzados em V_2 podem ser representados pela união dos eixos cruzados em quaisquer dois de seus vértices predecessores.

Esta propriedade pode ser observada no vértice 111 do hipercubo mostrado na figura 3.4. O cruzamento das informações de quaisquer dois de seu vértices predecessores (110, 011 e 101) basta para gerar suas informações.

Pela estrutura de dados representada pelo hipercubo, vê-se que este processo é facilmente paralelizável. Com a propagação das informações dando-se através de um hipercubo, todas as combinações de símbolos de predicado são avaliadas concorrentemente. Para paralelizar tal processo, cada vértice poderia ser representado por um processador e a propagação dar-se-ia via comunicação de processos. Esta é uma característica inerente ao algoritmo.

Porém, a distribuição da tarefa de combinação de literais não é a única justificativa para o uso do hipercubo. O algoritmo possui uma outra característica importante: gera diretamente o conjunto de implicantes primários de um conjunto de claúsulas.

A relação de subsunção entre cláusulas define uma ordem parcial para um conjunto de cláusulas. Esta ordem parcial está diretamente relacionada aos símbolos de predicado que compõem cada cláusula do conjunto. Ao ser realizada através de um hipercubo, a propagação das informações dá-se segundo uma ordem de precedência das mesmas. Como pode ser observado na figura 3.5, a ordem parcial estabelecida para as cláusulas duais do exemplo 7 é representada pelos caminhos da propagação do hipercubo.

Com isto tem-se garantido que se uma cláusula C_1 subsume uma cláusula C_2, C_2 pertence a um vértice sucessor de C_1. Também, pela propagação dar-se da menor dimensão para a maior, tem-se garantido que C_1 é gerada antes de C_2. Com isto, reconhecendo-se C_1, a geração de C_2 pode ser evitada. Na propagação descrita acima isto é feito quando, ao ser gerada uma cláusula dual, a combinação de literais que lhe deu origem não mais é propagada para os vértices sucessores. Se o fosse, toda a cláusula dual a qual esta combinação desse origem seria subsumida por aquela gerada no vértice predecessor.

[4]A diferença entre o número da dimensão de dois vértices é igual a diferença do número de eixos representados por estes.

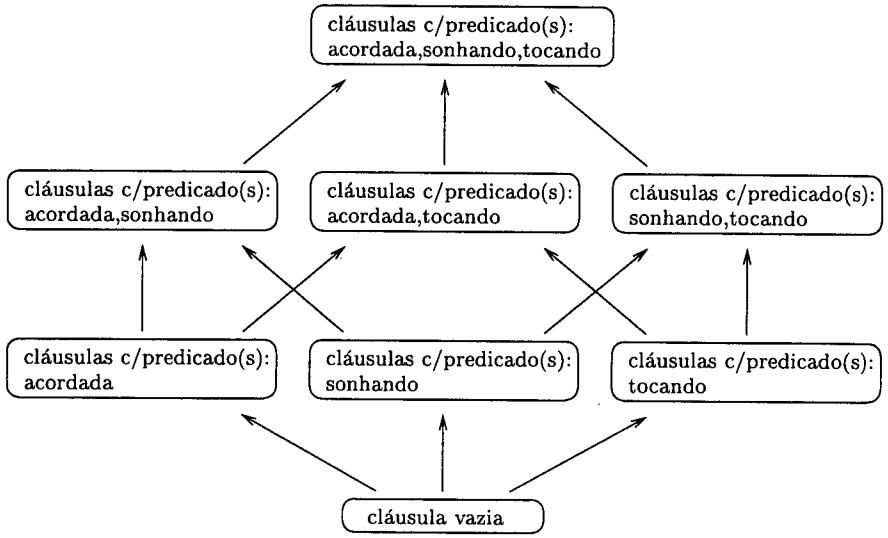

Figura 3.5: Ordem parcial para cláusulas duais do exemplo 7

A subsunção entre cláusulas duais é evitada pelo uso da estrutura de dados na forma de um hipercubo. Já a subsunção interna às cláusulas duais (entre seus literais) é evitada no instante da combinação entre os literais de mesmo predicado que ocorre nos vértices de primeira dimensão. Infelizmente, este fato não pode ser observado no exemplo utilizado para ilustar a dinâmica do algoritmo, pois o conjunto de cláusulas do exemplo não apresenta nenhuma relação de subsunção entre seus literais. Na descrição do algoritmo, feita a seguir, é apresentado o tratamento da subsunção entre literais.

3.2.2 O ALGORITMO

Com a dinâmica do algoritmo proposto já apresentada, apresenta-se a definição do algoritmo propriamente dito. Para apresentá-lo, o exemplo 18 é utilizado como referência.

Exemplo 18 *Considere a transformação do conjunto de cláusulas do exemplo 16. Suas formas clausal e clausal dual são reapresentadas:*

$$W_c = \langle\, [P(a), Q(a)],\, [P(a), R(x_1)],\, [P(x_2), R(a)]\,\rangle$$

$$W_d = [\, \langle P(x_2)\rangle,\, \langle P(a), R(a)\rangle,\, \langle Q(a), R(x_1)\rangle\,]$$

Este exemplo foi escolhido por abranger uma gama maior de detalhes importantes no processo de transformação proposto (quando algum detalhe não é coberto por este exemplo, um exemplo genérico é fornecido).

O algoritmo é dividido em três partes funcionais: o *Controle*, a *Inicialização* e o *Processamento Local*. Estas partes podem assim ser descritas:

- **Controle** (*Hologram-starting*[5])

  O controle é responsável (i) pela inicialização do processo de transformação dual, (ii) pela sincronização, a partir da primeira até a última dimensão do hipercubo, do processamento local de cada vértice do hipercubo de mesma dimensão e (iii) pela propagação das informações (*pacote de saída*) dos vértices para seus sucessores.

  O item (i) é realizado pela "chamada" ao processo de inicialização. O item (ii) pela "ativação" dos processos locais dos vértices de mesma dimensão, com o cuidado de nenhum processo local de um vértice de dimensão n ser ativado antes de todos os processos locais a vértices de dimensão *n-1* terem sido encerrados (e seus pacotes de saída propagados). Já, o item (iii) é realizado quando do término do processamento local dos vértices de dimensão n, pelo envio dos pacotes de saída destes vértices para seus vértices sucessores (o pacote de saída de um vértice é um *pacote de entrada* de seu sucessor).

  Apresenta-se a definição da rotina de controle:

  $Controle(W_c)$
  executar inicialização $(Ondas - iniciais(W_c))$
  para cada dimensão:
  para cada vertice $\mathcal{V}$ da dimensão:
  ativar o processamento local $(Dual - vertex(\mathcal{V}))$
  ao final do processamento local de todos os vertices da dimensão:
  para cada vertice da dimensão:
  propagar seu pacote de saida para seus vertices sucessores
  para cada vertice $\mathcal{V}$ do hipercubo:
  $W_d \leftarrow W_d \cup (Clausulas - duais(Particulas(\mathcal{V}))$

  onde **Cláusulas-duais (Partículas ($\mathcal{V}$))** na última linha, é uma função que transforma as partículas geradas no vértice $\mathcal{V}$ (se houverem) em cláusulas duais.

- **Inicialização** (*Initial-waves*)
  Pelo algoritmo proposto, o primeiro passo de transformação de um conjunto

[5]Nome do método implementado que representa esta parte funcional.

de cláusulas W_c para o conjunto de implicantes primários de seu conjunto de cláusulas duais equivalente W_d, consiste no cálculo das *Ondas Iniciais*. Para tal, antes é necessária a adaptação das informações de W_c para as estruturas de dados propostas no algoritmo.

Para o algoritmo um literal é uma unidade de informação denominada *quantum*. Estes quanta agrupados formam *ondas* que são propagadas através do hipercubo. A metáfora adotada visa facilitar o entendimento não tendo nenhuma outra conotação. Estas estruturas são assim definidas:

- **quantum**: representa um literal e seus atributos. Ele possui dois conjuntos numéricos de *coordenadas* de posição do literal no conjunto original de cláusulas. A um destes conjuntos chamamos *fix*, que corresponde aos números das cláusulas originais em que o literal aparece. Ao outro chamamos *sub*, que corresponde aos números das cláusulas originais nas quais aparecem literais subsumidos pelo literal do quantum. A cada literal correspondem dois quanta: o phi-quantum (ϕ) e o psi-quantum (ψ). O ϕ dá origem às claúsulas e o ψ às cláusulas duais (esta diferenciação é necessária para a correta definição das coordenadas dos quanta subsumidos). Um quantum que representa um literal que ocorre uma única vez no conjunto inicial e não subsume literal algum (ou seja, um quantum cujo *fix* tem apenas um elemento e cujo *sub* é um conjunto vazio) é chamado de *colapsado*. Caso contrário, é chamado de *aberto*. As coordenadas representadas no *sub* de um quantum evita a geração de cláusulas não puras;
- **onda**: representada por γ e notada por $\{\cdots\}$, é simplesmente um conjunto de quanta. Ela contém os literais que potencialmente originarão as novas cláusulas do estado dual. De sua formação e interação depende todo o processo de transformação. Uma onda que não possui quantum algum é uma *onda vazia* (notada por γ_0), aquela que possui apenas quanta colapsados é uma *onda colapsada*, senão é uma *onda aberta*. O objetivo da transformação dual é gerar ondas com um representante (quantum) de cada cláusula do conjunto original. Uma onda que contém pelo menos um número de coordenada (nos conjuntos *fix* ou *sub* de seus quanta) para cada cláusula é uma *onda saturada*.

Definidas as estruturas de dados pode-se apresentar as operações envolvidas neste primeiro passo. Primeiramente, numera-se o conjunto inicial de cláusulas e geram-se os quanta correspondentes (ver tabela 3.1).

Número	Cláusula	ψ quanta
0	$[P(a), Q(a)]$	$P(a)^{(\{0,1\},\{\})}, Q(a)^{(\{0\},\{\})}$
1	$[P(a), R(x_1)]$	$P(a)^{(\{0,1\},\{\})}, R(x_1)^{(\{1\},\{2\})}$
2	$[P(x_2), R(a)]$	$P(x_2)^{(\{2\},\{0,1\})}, R(a)^{(\{2\},\{\})}$

Tabela 3.1: Cláusulas do exemplo 18 e quanta associados

Separam-se os quanta iniciais em grupos de quanta de mesmo predicado. Para cada um destes grupos: (i) gera-se uma onda aberta para cada quantum aberto e uma onda colapsada com todos os quanta colapsados restantes; (ii) tenta-se combinar as ondas de todas as formas possíveis (cálculo do conjunto potência das ondas de mesmo grupo). Para cada subgrupo de ondas a combinar, aplica-se recursivamente uma operação binária denominada *interferência* definida da seguinte forma: dadas duas ondas γ_a e γ_b, seja γ_c a onda resultante da interferência entre as duas primeiras, tem-se:

- Se γ_a ou γ_b for uma onda vazia, então $\gamma_c = \gamma_0$.
- Se γ_a e γ_b são ondas colapsadas, então γ_c é uma onda formada pelos quanta de γ_a e γ_b (união das ondas).
- Se γ_a é uma onda aberta e γ_b é uma onda colapsada, então γ_c é uma onda formada pelos quanta de γ_a e os quanta de γ_b cujo *fix*, que é unico, não ocorra em nenhum *fix* ou *sub* de γ_a (γ_a acrescida dos quanta de γ_b que representam cláusulas do conjunto original antes não representadas em γ_a).
- Se γ_a e γ_b são ondas abertas, então:
  (i) se toda coordenada do *fix* de algum dos quanta abertos de γ_a ocorre no *fix* de outro (ou outros) quantum aberto de γ_b (ou vice-versa), ou, se a intersecção do *fix* de um quantum aberto de γ_a com o *sub* de algum quantum aberto de γ_b (ou vice-versa) não for vazia, então as duas ondas são incompatíveis e $\gamma_c = \gamma_0$.
  (ii) senão, as ondas são compatíveis e γ_c é uma onda formada pelos quanta abertos de γ_a e γ_b e pelos quanta colapsados de γ_a e γ_b cujo *fix* não ocorra em nenhum *fix* ou *sub* dos quanta abertos de γ_c.

Exemplo 19 *Para as cláusulas do exemplo 18, os grupos de ondas iniciais são:*

$$(\{\neg P(a)^{(\{0,1\},\{\})}\}, \{P(x_2)^{(\{2\},\{0,1\})}\}),$$

$$(\{Q(a)^{(\{0\},\{\})}\}) \; e$$

$$(\{R(x_1)^{(\{1\},\{2\})}\}, \{R(a)^{(\{2\},\{\})}\})$$

Nota-se que nenhuma combinação entre ondas do mesmo grupo foi bem sucedida. Para ilustrar um caso onde isto acontece, apresenta-se o processo de criação das onda iniciais a partir do seguinte grupo de quanta:

$$(\neg P(x_1)^{(\{2\},\{0\})}, \; P(x_2)^{(\{1\},\{\})}, P(x_3)^{(\{2\},\{\})}, \; \neg P(a)^{(\{0\},\{\})})$$

com as seguintes ondas iniciais como resultado:

- $\{\neg P(x_1)^{(\{2\},\{0\})}\}$ *onda aberta gerada a partir do quantum aberto;*
- $\{P(x_3)^{(\{2\},\{\})}, P(x_2)^{(\{1\},\{\})}, \neg P(a)^{(\{0\},\{\})}\}$ *onda colapsada gerada a partir dos quanta colapsados;*
- $\{P(x_2)^{(\{1\},\{\})}, \neg P(x_1)^{(\{2\},\{0\})}\}$ *onda aberta resultante da interferência entre as outras duas.*

O importante é ter-se em mente que a interferência entre duas ondas produz sempre uma onda válida, nem que esta seja vazia. Para uma onda ser válida, duas propriedades devem ser observadas:

- Todo o quantum da onda tem em seu *fix* pelo menos uma coordenada que não ocorre no *fix* de outro quantum. Ou seja, cada quantum da onda representa, com exclusividade, pelo menos uma cláusula do conjunto original.
- Nenhuma coordenada que aparece no *sub* de algum quantum da onda, aparece também em algum *fix* de outro quantum da mesma onda. Ou seja, toda cláusula do conjunto original representada no *fix* de um quantum da onda, não é subsumida (representada no *sub*) por outro quantum da mesma onda.

Baseando-se nas definições acima, apresenta-se a definição da rotina de inicialização (cálculo das ondas iniciais para os vértices de primeira dimensão):

$Ondas - Iniciais(W_c)$
01. $\mathcal{Q} \leftarrow$ (Psi-quanta(W_c))
02. para cada simbolo $\mathcal{P}$ de predicado:
03. $\quad \mathcal{Q}_p \leftarrow$ (Quanta($\mathcal{Q}, \mathcal{P}$))
04. $\quad \mathcal{W} \leftarrow$ Wave (Collapsed-quanta($\mathcal{Q}_p$)))

05. para cada quantum ψ de (Open-quanta($\mathcal{Q}_p$)):
06. $\mathcal{W} \leftarrow \mathcal{W} \cup$(Wave($\{\psi\}$))
07. $\mathcal{I} \leftarrow \emptyset$
08. para cada subconjunto $\int$ de (Power-set($\mathcal{W}$)):
09. se (Interference ($\int$)) $\neq \emptyset$
10. $\mathcal{I} \leftarrow \mathcal{I} \cup$ (Interference ($\int$))
11. Input-pack (Vertice ($\mathcal{P}$)) $\leftarrow \mathcal{I}$

onde:

- **psi-quanta** (W_c) na linha 1, é uma função que retorna o conjunto de quanta do tipo psi (ψ) associado ao conjunto de cláusulas W_c;
- **quanta**($\mathcal{Q}, \mathcal{P}$) na linha 3, é uma função que retorna o subconjunto de quanta de $\mathcal{Q}$ que representam literais de símbolo de predicado igual a $\mathcal{P}$;
- **wave (Q))** na linha 4, é uma função que retorna uma onda formada pelo conjunto de quanta Q;
- **collapsed-quanta** ($\mathcal{Q}_p$) na linha 4, é uma função que retorna o subconjunto de quanta colapsados de $\mathcal{Q}_p$;
- **open-quanta** ($\mathcal{Q}_p$) na linha 5, é uma função que retorna o subconjunto de quanta abertos de $\mathcal{Q}_p$;
- **power-set** ($\mathcal{W}$) na linha 8, é uma função que retorna um conjunto das combinações dos elementos de $\mathcal{W}$ (também chamado de conjunto de partes de $\mathcal{W}$);
- **input-pack (V)** na linha 11, é o conjunto de pacotes de entrada do vértice V;
- **vertice** ($\mathcal{P}$) na linha 11, é o vértice (de primeira dimensão) associado ao predicado $\mathcal{P}$.

- **Processamento Local** (*Dual-vertex*)
Ao término do cálculo das ondas iniciais tem-se, em cada vértice de primeira dimensão, as informações necessárias para a realização do processamento local de cada vértice. Este processamento consiste em: combinar as informações de entrada, verificar se alguma cláusula dual foi gerada e, por fim, calcular as informações de saída que devem ser propagadas. Apresentamos as estruturas de dados envolvidas neste processamento local.

Para o algoritmo, em cada vértice do *holograma*, encontra-se uma *órbita*. É nesta órbita que as ondas transitam (chegam nos pacotes de entrada, chocam-se e são enviadas através do pacote de saída). Nesta órbita, também, uma onda saturada pode dar origem a uma ou mais *partículas*. Estas estruturas são assim definidas:

- **holograma**: representa o hipercubo em si. Cada um de seus eixos de primeira dimensão é associado a um número (iniciando em zero) e representa um símbolo de predicado que ocorre no conjunto original de cláusulas. Cada vértice de dimensão n representa os cruzamentos das informações de n eixos de dimensão 1 e é numerado com a soma das potências de 2 de cada eixo representado[6] (este número é a representação decimal do número binário indicado pela *tarja* do vértice). Todo vértice de dimensão n é *sucessor* dos vértices de dimensão $n-1$ cujos eixos representados por estes sejam também representados por aquele (o oposto pode ser usado para definir os vértices *predecessores* de um vértice). O holograma para o exemplo 16 está ilustrado na figura 3.6;
- **órbita**: contém os dados locais de um vértice do holograma. Em cada vértice "reside" uma órbita com as seguintes informações: o número do vértice, os "pacotes" de ondas de entrada (um para cada órbita predecessora), as partículas geradas e o pacote de ondas de saída (único por órbita). Estes pacotes são explicados a seguir;
- **partícula**: é uma onda estática, que não é propagada. Uma onda saturada dá origem a uma ou mais partículas. As partículas representam as cláusulas duais de W_d resultantes do processo de transformação dual de W_c.

Em cada órbita dá-se o seguinte processamento local:

- Se algum dos pacotes de entrada é vazio, é porque o cruzamento das informações dos eixos representados por alguma das órbitas predecessoras não obteve sucesso (e, certamente, nenhuma órbita sucessora a esta obterá tal êxito). Deve-se, pois, interromper o processamento local. Com esta interrupção o pacote de saída da órbita permanece vazio, causando a interrupção em cadeia do processamento local das órbitas sucessoras.
- Senão os pacotes de entrada são combinados entre si pegando-se uma onda de cada pacote. Em cada combinação aplica-se, como nas ondas iniciais, a interferência.

[6] Por exemplo, o vértice que representa o cruzamento dos eixos 0 e 1 recebe o número $(2^0 + 2^1) = 3$.

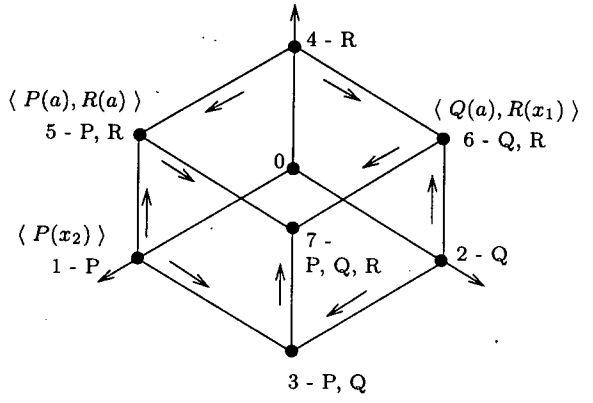

Figura 3.6: Holograma associado ao exemplo 18

Como uma órbita necessita de apenas dois de seus pacotes de entrada (oriundos de duas órbitas predecessoras quaisquer), uma boa estratégia é combinar ondas provenientes somente dos dois menores pacotes de entrada, o que pode reduzir consideravelmente o número de combinações a serem avaliadas. Em geral, os dois primeiros pacotes a chegarem têm uma boa possibilidade de serem também os menores, permitindo uma estratégia de processamento oportunístico. Deve-se observar, no entanto, que antes que o resultado seja enviado aos sucessores, deve-se aguardar a chegada dos resultados do processamento local de todas órbitas predecessoras, pois pode acontecer do pacote de saída de uma delas ser vazio. Em órbitas de primeira dimensão, este passo não é necessário já que as ondas iniciais já foram combinadas.

Para cada onda nova gerada (ou para cada onda inicial, se órbita de primeira dimensão):

- Se a onda for vazia, desconsiderar.
- Se a onda for saturada, gerar as partículas distribuindo-se os quanta desta onda de modo que todas as cláusulas originais sejam representadas (em *fix* ou *sub*) uma única vez em cada cláusula dual (ver exemplo 20). Proceder a *reemissão* desta onda: para cada quantum da onda, gerar uma nova onda sem ele; se a onda resultante for desta órbita, colocá-la no pacote de saída da órbita.
- Senão, colocar a onda no pacote de saída da órbita.

Dim	Vert	Pacotes de Entrada	Cláusulas duais	Pacote de Saída
1	1	$(\{P(a)^{(\{0,1\},\{\})}\},$ $\{P(x_2)^{(\{2\},\{0,1\})}\})$	$\langle P(x_2)\rangle$	$(\{P(a)^{(\{0,1\},\{\})}\})$
	2	$(\{Q(a)^{(\{0\},\{\})}\})$		$(\{Q(a)^{(\{0\},\{\})}\})$
	4	$(\{R(x_1)^{(\{1\},\{2\})}\},$ $\{R(a)^{(\{2\},\{\})}\})$		$(\{R(x_1)^{(\{1\},\{2\})}\},$ $\{R(a)^{(\{2\},\{\})}\})$
2	3	$(\{P(a)^{(\{0,1\},\{\})}\}),$ $(\{Q(a)^{(\{0\},\{\})}\})$		$()$
	5	$(\{P(a)^{(\{0,1\},\{\})}\}),$ $(\{R(x_1)^{(\{1\},\{2\})}\},$ $\{R(a)^{(\{2\},\{\})}\})$	$\langle R(a), P(a)\rangle$	$()$
	6	$(\{Q(a)^{(\{0\},\{\})}\}),$ $(\{R(x_1)^{(\{1\},\{2\})}\},$ $\{R(a)^{(\{2\},\{\})}\})$	$\langle Q(a), R(x_1)\rangle$	$(\{Q(a)^{(\{0\},\{\})},$ $R(a)^{(\{2\},\{\})}\})$
3	7	$(), (),$ $(\{Q(a)^{(\{0\},\{\})},$ $R(a)^{(\{2\},\{\})}\})$		

Tabela 3.2: Informações das órbitas no processamento local

Exemplo 20 *Uma onda saturada pode dar origem a uma ou mais partículas. Considerando-se um conjunto dado de cláusulas, onde a onda:*

$$\{P^{(\{0\},\{\})}, Q^{(\{0\},\{\})}, R^{(\{1\},\{\})}, S^{(\{2\},\{\})}, T^{(\{3\},\{\})}, U^{(\{4\},\{\})}\}$$

aparece como uma das ondas saturadas do processo de transformação, gera-se a partir dela as partículas:

$$\{P^{(\{0\},\{\})}, R^{(\{1\},\{\})}, S^{(\{2\},\{\})}, T^{(\{3\},\{\})}, U^{(\{4\},\{\})}\}$$

$$\{Q^{(\{0\},\{\})}, R^{(\{1\},\{\})}, S^{(\{2\},\{\})}, T^{(\{3\},\{\})}, U^{(\{4\},\{\})}\}$$

onde, em cada uma, um literal diferente foi escolhido como representante da cláusula 0 (em uma partícula a cláusula 0 é representada pelo quantum $P^{(\{0\},\{\})}$ *e na outra pelo quantum* $Q^{(\{0\},\{\})}$*, dando assim origem a duas cláusulas duais que diferem apenas pelo primeiro literal). Note-se que, pelas coordenadas que ocorrem em uma onda saturada sabe-se o número de cláusulas do conjunto inicial. Neste exemplo o conjunto original contém 5 cláusulas, já que as coordenadas variam de 0 a 4.*

Deste processamento resultam zero ou mais partículas geradas na órbita e zero ou mais ondas no pacote de saída da mesma.

Após o término do processamento local das órbitas de mesma dimensão, a função de controle reassume o processamento. Através dela os pacotes de saída destas órbitas são propagados para os pacotes de entrada das órbitas sucessoras e é reiniciado o ciclo de processamento local em cada órbita da nova dimensão. Este procedimento se repete até ter-se completado o processamento local das órbitas de última dimensão do holograma.

Exemplo 21 *Para o exemplo 18, os caminhos da propagação e as cláusulas duais resultantes são ilustrados através das setas da figura 3.6. As informações de cada órbita do holograma durante o processo de propagação aparecem na tabela 3.2. Observa-se que o processamento local da órbita do vértice de número 7 foi interrompido por existirem pacotes vazios entre os pacotes de entrada da mesma.*

No final do processamento local de todas as órbitas do holograma, a função de controle "recolhe" as partículas nelas geradas e as transforma em cláusulas duais dando origem a W_d. É interessante notar estas cláusulas são geradas baseadas apenas nas informações locais de cada órbita.

Baseando-se nas definições dadas, apresenta-se a definição de algoritmo para o processamento local da órbita de um vértice $\mathcal{V}$:

$processo - local(\mathcal{V})$
01. se input-pack($\mathcal{V}$) não contem nenhum pacote vazio
02. $\mathcal{W} \leftarrow$ (Interference (Distr (input-pack($\mathcal{V}$))))
03. $\mathcal{N} \leftarrow$ (Join-Collapsed (New-waves ($\mathcal{W}$)))
04. output-pack($\mathcal{V}$) $\leftarrow \emptyset$
05. particles($\mathcal{V}$) $\leftarrow \emptyset$
06. para cada onda γ de $\mathcal{N}$:
07. se $\gamma = \gamma_0$, desconsiderar
08. se γ e' saturada:
09. particles($\mathcal{V}$) $\leftarrow$ $particles(\mathcal{V}) \cup (Create - particles(\gamma))$
10. se $\mathcal{V}$ não e' vertice de ultima dimensão:
11. output-pack($\mathcal{V}$) $\leftarrow$ output-pack($\mathcal{V}$) $\cup$ $(Reemission(\gamma))$
12. se $\mathcal{V}$ não e' vertice de ultima dimensão:
13. output-pack($\mathcal{V}$) $\leftarrow$ output-pack($\mathcal{V}$) $\cup \{\gamma\}$

onde:

- **distr (input-pack($\mathcal{V}$))** na linha 2, é uma função que seleciona os dois menores pacotes de input-pack($\mathcal{V}$) e combina suas ondas, gerando pares de ondas formados por uma onda de cada pacote;
- **new-waves ($\mathcal{N}$)** na linha 3, é uma função que seleciona, de um conjunto de ondas, aquelas que pertencem ao vértice $\mathcal{V}$ (além de eliminar possíveis duplicidades de ondas que possam ocorrer em $\mathcal{N}$);
- **join-Collapsed (W)** na linha 3, é uma função que "une", via interferência, todas ondas colapsadas de W em uma só onda colapsada. O retorno da função é o conjunto de ondas W atualizado;
- **create-particles (γ)** é uma função cujo retorno é o conjunto de partículas geradas a partir da onda saturada γ;
- **reemision (γ)** é uma função cujo retorno é o conjunto das ondas, do vértice $\mathcal{V}$, geradas pela reemissão de uma onda saturada.

O cuidado em evitar que alguns passos sejam executados no vértice da última dimensão, serve apenas para otimizar o processamento. Como este vértice não possui sucessores, seu pacote de saída não precisa conter informação alguma no final de seu processamento local.

O algoritmo proposto é totalmente simétrico para o caso dual.

3.3 CONCLUSÃO

Foi apresentado um algoritmo para realizar a transformação entre as formas normais canônicas conjuntiva e disjuntiva da lógica de primeira ordem, de maneira eficiente. Este algoritmo possui duas características fundamentais:

- ele gera, a partir de um conjunto de cláusulas (duais), diretamente o conjunto de implicantes primários de seu conjunto dual, sem gerar cláusulas (duais) subsumidas e,
- sua estrutura o torna naturalmente concorrente e, por isto, facilmente paralelizável.

Antes de sua apresentação, abordou-se características e relações mantidas entre as duas formas normais canônicas de representação de uma fórmula lógica. Outros algoritmos propostos para o mesmo fim foram citados.

Colocado o problema da transformação dual, foi inicialmente abordada a dinâmica do algoritmo proposto. A dinâmica foi apresentada passo a passo para proporcionar um entendimento intuitivo de como se dá a transformação dual pelo algoritmo proposto.

O algoritmo propriamente dito foi apresentado a seguir. Para sua descrição foi priorizado o entendimento de suas características e propriedades, nem sempre claras em uma formalização. O algoritmo apresentado, apesar de algumas pequenas adaptações, segue estritamente sua definição formal. Estas adaptações restringem-se basicamente às estruturas de dados de entrada e saída do algoritmo - as cláusulas e cláusulas duais - que no artigo de origem são tratadas como estruturas matemáticas genéricas. Outras pequenas adaptações devem-se à evolução do algoritmo ao longo do desenvolvimento deste trabalho.

Capítulo 4

A Implementação

Neste capítulo apresenta-se o ambiente onde o *Método de Inferência baseado na Transformação Dual*, definido nos capítulos anteriores, foi implementado. Exemplo de interação com este ambiente são fornecidos com intuito de mostrar como o mesmo pode ser utilizado como uma ferramenta para o estudo e desenvolvimento das potencialidades daquele método.

4.1 O LOGIK

O LOGIK [TB96] é um "laboratório" orientado a objeto para lógica de primeira ordem, escrito na linguagem de programação Common Lisp utilizando CLOS (do inglês, "Common Lisp Object System"), a extensão orientada objeto padrão do Common Lisp.

Nele estão incluídos o *Método de Inferência baseado na Transformação Dual* e o algoritmo descrito para a transformação dual propriamente dita, além da operação de *Box* (não tratada neste trabalho) que é um dos passos da definição da inferência. Todas as entidades lógicas - variáveis, funções, predicados, termos, literais, cláusulas, cláusulas duais e substituições - foram implementadas como classes com seus métodos associados (como aplicação da substituição, unificação e subsunção, por exemplo).

Fazendo uso destas classes já definidas, os referidos método e algoritmo foram implementados, estendendo assim o ambiente LOGIK. Para tal, novas classes específicas como quantum, onda, partícula, teoria e holograma foram acrescidas. Estas classes já foram apresentadas nos capítulos anteriores. Além destas, outra importante classe foi definida:

- **vernáculo**: é um tabela do tipo "hash-table". Suas entradas são indexadas por predicado, sinal e matriz do literal. O sintagma é o conteúdo de cada uma destas entradas. A cada literal corresponde um sintagma. Cada sintagma contém a lista

dos literais subsumidos por este, dos literais que o subsumem, dos literais complementares e das variantes alfabéticas[1] e os quanta equivalentes. Este mecanismo de representação explícita da subsunção entre os literais pode parecer "caro", mas ele ocorre uma única vez para cada literal incluído no vernáculo e a indexação da entrada deste reduz o espaço de busca (limitada aos literais de mesmos predicado e sinal) além de auxiliar em muito durante o cálculo do conjunto dual. Cada literal é incluído uma única vez no vernáculo independentemente do número de teorias em que ele ocorre.

4.2 A INTERAÇÃO COM O AMBIENTE

Como exemplo a ser analisado, apresenta-se as entradas e saídas da definição das seguintes teorias:

- *Nat* que apresenta uma definição para os números naturais.
- *Plus* que apresenta uma definição para a soma de dois números naturais.
- *P2* que define um teorema a ser provado baseado nas duas teorias anteriores. Com este teorema quer-se saber qual a soma de 2 + 1.

A primeira primitiva definida é denominada **tell**. Ela é responsável por receber um conjunto de cláusulas e transformá-lo em uma teoria.

```
> (tell '(((Nat 0)) ((Not (Nat X)) (Nat (Suc X))))
        'nat)

**********************************************************
theory: Nat

particles:
(1) - 0:{Nat(o)({0 1},{})}
(1) - 1:{\~{N}at(?x-1)({1},{}),Nat(suc(?x-1))({0},{})}

dual-particles:
(1) - 0:{Nat(o)({0},{}),Nat(suc(?x-1))({1},{})}
(1) - 1:{Nat(o)({0},{}),\~{N}at(?x-1)({1},{})}
```

[1] Literais que diferem apenas pelo nome das variáveis.

```
**********************************************************
>
```

Percebe-se que para a teoria ser criada, a transformação do conjunto de cláusulas de entrada foi convertido para um conjunto de partículas e foi calculado o seu conjunto de partículas duais. Estas partículas duais representam as cláusulas duais resultantes da aplicação da operação de transformação dual no conjunto de cláusulas originais. Esta transformação pode também ser acompanhada no ambiente (como na teoria é mencionado apenas um símbolo de predicado, toda a transformação se dá em uma única órbita):

```
==============================================
Orbit vertex 1 (dimension 1)

input packs:
(1) : {Nat(suc(?x-1))({1},{}),\~{N}at(?x-1)({1},{}),Nat(o)({0},{})}


emited particles:
(1) : {Nat(o)({0},{}),\~{N}at(?x-1)({1},{})}
(1) : {Nat(o)({0},{}),Nat(suc(?x-1))({1},{})}


output pack:


**********************************************************
```

No ambiente, a concorrência dos processos locais às órbitas foi apenas simulada. Estes processos, na realidade ocorrem seqüencialmente no ambiente. Conforme visto, para cada órbita, são informados: o número do vértice, a dimensão do hipercubo a qual a órbita pertence, as ondas que compõem os pacotes de entrada, as partículas emitidas e as ondas que compõem o pacote de saída da órbita.

Segue-se a informação das outras teorias (mostrando-se como se dá o processo de transformação dual):

```
> (tell '(((Not (Nat X)) (Plus 0 X X)) ((Not (Plus X Y Z))
```

```
          (Plus (Suc X) Y (Suc Z))))
         'plus)
================================================
Orbit vertex 2 (dimension 1)

input packs:
(1 0) : {Plus(suc(?x-2), ?x-3, suc(?x-4))({1},{}),
         ~Plus(?x-2, ?x-3, ?x-4)({1},{}),Plus(o, ?x-1, ?x-1)({0},{})}


emited particles:
(1 0) : {Plus(o, ?x-1, ?x-1)({0},{}),
         ~Plus(?x-2, ?x-3, ?x-4)({1},{})}
(1 0) : {Plus(o, ?x-1, ?x-1)({0},{}),
         Plus(suc(?x-2), ?x-3, suc(?x-4))({1},{})}


output pack:
(1 0) : {Plus(suc(?x-2), ?x-3, suc(?x-4))({1},{}),
         ~Plus(?x-2, ?x-3, ?x-4)({1},{})}
(1 0) : {Plus(o, ?x-1, ?x-1)({0},{})}


================================================
Orbit vertex 1 (dimension 1)

input packs:
(0 1) : {\~{N}at(?x-1)({0},{})}


emited particles:


output pack:
(0 1) : {\~{N}at(?x-1)({0},{})}
```

```
==============================================
Orbit vertex 3 (dimension 2)

input packs:
(0 1) : {\~{N}at(?x-1)({0},{})}

(1 0) : {Plus(suc(?x-2), ?x-3, suc(?x-4))({1},{}),
         ~Plus(?x-2, ?x-3, ?x-4)({1},{})}
(1 0) : {Plus(o, ?x-1, ?x-1)({0},{})}


emited particles:
(1 1) : {\~{N}at(?x-1)({0},{}),~Plus(?x-2, ?x-3, ?x-4)({1},{})}
(1 1) : {\~{N}at(?x-1)({0},{}),Plus(suc(?x-2), ?x-3, suc(?x-4))({1},{})}


output pack:


************************************************************
theory: Plus

particles:
(1 1) - 0:{\~{N}at(?x-1)({2 3},{}),Plus(o, ?x-1, ?x-1)({0 1},{})}
(1 0) - 1:{~Plus(?x-2, ?x-3, ?x-4)({1 3},{}),
           Plus(suc(?x-2), ?x-3, suc(?x-4))({0 2},{})}

dual-particles:
(1 0) - 0:{Plus(o, ?x-1, ?x-1)({0},{}),
          Plus(suc(?x-2), ?x-3, suc(?x-4))({1},{})}
(1 0) - 1:{Plus(o, ?x-1, ?x-1)({0},{}),~Plus(?x-2, ?x-3, ?x-4)({1},{})}
(1 1) - 2:{\~{N}at(?x-1)({0},{}),Plus(suc(?x-2), ?x-3, suc(?x-4))({1},{})}
(1 1) - 3:{\~{N}at(?x-1)({0},{}),~Plus(?x-2, ?x-3, ?x-4)({1},{})}
************************************************************
> (tell '(((Not (Plus (Suc (Suc 0)) (Suc 0) X))))
        'p2)
```

```
==============================================
Orbit vertex 2 (dimension 1)

input packs:
(1 0) : {~Plus(suc(suc(o)), suc(o), ?x-1)({0},{})}


emited particles:
(1 0) : {~Plus(suc(suc(o)), suc(o), ?x-1)({0},{})}


output pack:


************************************************************
theory: P2

particles:
(1 0) - 0:{~Plus(suc(suc(o)), suc(o), ?x-1)({0},{})}

dual-particles:
(1 0) - 0:{~Plus(suc(suc(o)), suc(o), ?x-1)({0},{})}
************************************************************
>
```

Informadas as três teorias ao ambiente, pode-se querer uni-las, via operação de **box** a fim de formar uma única teoria para ser utilizada para a prova.

```
> (box '(nat plus p2) 'soma2+1)

************************************************************
theory: Soma2+1

particles:
(0 1) - 0:{Nat(o)({0 1 2 3 4 5 6 7},{})}
(0 1) - 1:{\~{N}at(?x-1)({0 2 5 7},{}),Nat(suc(?x-1))({1 3 4 6},{})}
(1 1) - 2:{\~{N}at(?x-2)({0 1 6 7},{}),Plus(o, ?x-2, ?x-2)({2 3 4 5},{})}
(1 0) - 3:{~Plus(?x-5, ?x-4, ?x-3)({0 1 2 3},{}),
```

```
            Plus(suc(?x-5), ?x-4, suc(?x-3))({4 5 6 7},{})}
(1 0) - 4:{~Plus(suc(suc(o)), suc(o), ?x-6)({4 5 6 7},{0 1 2 3})}

dual-particles:
(1 1) - 0:{\~{N}at(?x-2)({2},{}),\~{N}at(?x-1)({1},{}),Nat(o)({0},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 1:{\~{N}at(?x-2)({2},{}),Nat(suc(?x-1))({1},{}),Nat(o)({0},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 2:{Plus(o, ?x-2, ?x-2)({2},{}),\~{N}at(?x-1)({1},{}),Nat(o)({0},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 3:{Plus(o, ?x-2, ?x-2)({2},{}),Nat(suc(?x-1))({1},{}),Nat(o)({0},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 4:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
           Plus(o, ?x-2, ?x-2)({2},{}),
           Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{}),Nat(o)({0},{}),
           Nat(suc(?x-1))({1},{})}
(1 1) - 5:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
           Plus(o, ?x-2, ?x-2)({2},{}),
           Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{}),
           Nat(o)({0},{}),\~{N}at(?x-1)({1},{})}
(1 1) - 6:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
           \~{N}at(?x-2)({2},{}),Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{}),
           Nat(o)({0},{}),Nat(suc(?x-1))({1},{})}
(1 1) - 7:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
           \~{N}at(?x-2)({2},{}),Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{}),
           Nat(o)({0},{}),\~{N}at(?x-1)({1},{})}
**********************************************************
```

Criada a teoria, pode-se utilizar a rotina *Pulsar* para provar que a mesma é contraditória (como está-se utilizando a rotina pulsar sem estratégias, a mesma foi denominada **Saturation** em alusão à estratégia de saturação de nível (do inglês "Level Saturation") do método de Resolução).

```
> (saturation 'soma2+1)
```

que após 17 teorias geradas, resultantes da aplicação de 17 substituições diferentes que tornam a teoria original contraditória, um novo ciclo de inferência é iniciado com a seguinte teoria gerada:

```
**********************************************************
theory: New

particles:
(0 1) - 0:{Nat(o)({0 1 2 3 4 5 6 7 8 9 10 11},{})}
(0 1) - 1:{\~{N}at(?x-1)({0 2 5 7 9 11},{}),Nat(suc(?x-1))({1 3 4 6 8 10},{})}
(1 1) - 2:{\~{N}at(?x-2)({0 1 6 7 10 11},{}),
          Plus(o, ?x-2, ?x-2)({2 3 4 5 8 9},{})}
(1 0) - 3:{~Plus(?x-5, ?x-4, ?x-3)({4 5 6 7 8 9 10 11},{}),
          Plus(suc(?x-5), ?x-4, suc(?x-3))({0 1 2 3},{})}
(1 0) - 4:{~Plus(suc(suc(o)), suc(o), ?x-6)({0 1 2 3},{4 5 6 7 8 9 10 11})}
(1 0) - 5:{~Plus(suc(o), suc(o), ?x-7)({0 1 2 3},{4 5 6 7 8 9 10 11})}
(1 0) - 6:{Plus(o, o, o)({0 1 6 7 10 11},{2 3 4 5 8 9})}
(0 1) - 7:{Nat(suc(o))({0 2 5 7 9 11},{1 3 4 6 8 10})}
(1 0) - 8:{Plus(suc(o), o, suc(o))({8 9 10 11},{4 5 6 7 0 1 2 3})}
(1 0) - 9:{Plus(o, suc(o), suc(o))({0 1 6 7 10 11},{2 3 4 5 8 9})}
(1 0) - 10:{Plus(suc(o), suc(o), suc(suc(o)))({8 9 10 11},{4 5 6 7 0 1 2 3})}
(1 1) - 11:{\~{N}at(?x-8)({8 9 10 11},{}),
           Plus(suc(o), ?x-8, suc(?x-8))({4 5 6 7},{0 1 2 3})}

dual-particles:
(1 1) - 0:{Nat(o)({0},{}),\~{N}at(?x-1)({1},{}),\~{N}at(?x-2)({2},{}),
          ~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
          ~Plus(suc(o), suc(o), ?x-7)({5},{}),Plus(o, o, o)({6},{}),
          Nat(suc(o))({7},{}),Plus(o, suc(o), suc(o))({9},{}),
          Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{11 10 8})}
(1 1) - 1:{Nat(o)({0},{}),\~{N}at(?x-2)({2},{}),
          ~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
          ~Plus(suc(o), suc(o), ?x-7)({5},{}),Plus(o, o, o)({6},{}),
          Plus(o, suc(o), suc(o))({9},{}),
          Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{11 10 8}),
          Nat(suc(?x-1))({1},{7})}
(1 1) - 2:{Nat(o)({0},{}),\~{N}at(?x-1)({1},{}),
          ~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
          ~Plus(suc(o), suc(o), ?x-7)({5},{}),Nat(suc(o))({7},{}),
          Plus(o, ?x-2, ?x-2)({2},{9 6}),
          Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{11 10 8})}
```

```
(1 1) - 3:{Nat(o)({0},{}),~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
           ~Plus(suc(o), suc(o), ?x-7)({5},{}),
           Plus(o, ?x-2, ?x-2)({2},{9 6}),
           Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{11 10 8}),
           Nat(suc(?x-1))({1},{7})}
(1 1) - 4:{Nat(o)({0},{}),Plus(o, ?x-2, ?x-2)({2},{9 6}),
           Nat(suc(?x-1))({1},{7}),~Plus(?x-5, ?x-4, ?x-3)({3},{5 4}),
           Plus(suc(o), ?x-8, suc(?x-8))({11},{8 10})}
(1 1) - 5:{Nat(o)({0},{}),\~{N}at(?x-1)({1},{}),Nat(suc(o))({7},{}),
           Plus(o, ?x-2, ?x-2)({2},{9 6}),~Plus(?x-5, ?x-4, ?x-3)({3},{5 4}),
           Plus(suc(o), ?x-8, suc(?x-8))({11},{8 10})}
(1 1) - 6:{Nat(o)({0},{}),\~{N}at(?x-2)({2},{}),Plus(o, o, o)({6},{}),
           Plus(o, suc(o), suc(o))({9},{}),Nat(suc(?x-1))({1},{7}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{5 4}),
           Plus(suc(o), ?x-8, suc(?x-8))({11},{8 10})}
(1 1) - 7:{Nat(o)({0},{}),\~{N}at(?x-1)({1},{}),\~{N}at(?x-2)({2},{}),
           Plus(o, o, o)({6},{}),Nat(suc(o))({7},{}),
           Plus(o, suc(o), suc(o))({9},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{5 4}),
           Plus(suc(o), ?x-8, suc(?x-8))({11},{8 10})}
(1 1) - 8:{Nat(o)({0},{}),Plus(suc(o), o, suc(o))({8},{}),
           Plus(suc(o), suc(o), suc(suc(o)))({10},{}),\~{N}at(?x-8)({11},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{5 4}),Nat(suc(?x-1))({1},{7}),
           Plus(o, ?x-2, ?x-2)({2},{9 6})}
(1 1) - 9:{\~{N}at(?x-1)({1},{}),Nat(o)({0},{}),Nat(suc(o))({7},{}),
           Plus(suc(o), o, suc(o))({8},{}),
           Plus(suc(o), suc(o), suc(suc(o)))({10},{}),\~{N}at(?x-8)({11},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{5 4}),Plus(o, ?x-2, ?x-2)({2},{9 6})}
(1 1) - 10:{\~{N}at(?x-2)({2},{}),Nat(o)({0},{}),Plus(o, o, o)({6},{}),
            Plus(suc(o), o, suc(o))({8},{}),Plus(o, suc(o), suc(o))({9},{}),
            Plus(suc(o), suc(o), suc(suc(o)))({10},{}),\~{N}at(?x-8)({11},{}),
            ~Plus(?x-5, ?x-4, ?x-3)({3},{5 4}),Nat(suc(?x-1))({1},{7})}
(1 1) - 11:{\~{N}at(?x-2)({2},{}),\~{N}at(?x-1)({1},{}),Nat(o)({0},{}),
            Nat(suc(o))({7},{}),Plus(o, o, o)({6},{}),
            Plus(suc(o), o, suc(o))({8},{}),Plus(o, suc(o), suc(o))({9},{}),
            Plus(suc(o), suc(o), suc(suc(o)))({10},{}),\~{N}at(?x-8)({11},{}),
            ~Plus(?x-5, ?x-4, ?x-3)({3},{5 4})}
```

```
**********************************************************

Finding contradictions...

theorem of the theory Soma2+1:
. substitution:
{(?x-6/suc(?x-3)), (?x-3/suc(?x-8)), (?x-5/suc(o)), (?x-8/suc(o)),
 (?x-4/suc(o)), (?x-2/suc(o)), (?x-1/o)}
.[~Plus(suc(suc(o)), suc(o), suc(suc(suc(o))))]
```

Reiniciado o ciclo de inferência com a nova teoria, foi alcançada a contradição. O teorema com a substituição é mostrado (outras substituições foram omitidas). Percebe-se pelo estado final do teorema, a valor da soma de *suc(suc(o))* com *suc(o)* é *suc(suc(suc(o)))*, ou seja, $2 + 1 = 3$.

Para exemplificar o poder de uma estratégia, apresenta-se uma que está sendo estudada. A mesma baseia-se na tentativa de contradizer o teorema a cada ciclo de inferência, na tentativa de gerar uma cláusula vazia que implica na geração de um conjunto de cláusulas duais vazio (ou seja, a contradição). Esta estratégia foi denominada **pulsar**. Vamos mostrá-la utilizando a mesma teoria *soma2+1*.

```
> (pulsar 'soma2+1)

**********************************************************
theory: Soma2+1

particles:
(0 1) - 0:{Nat(o)({7 6 5 4 3 2 1 0},{})}
(0 1) - 1:{\~{N}at(?x-1)({7 5 2 0},{}),Nat(suc(?x-1))({6 4 3 1},{})}
(1 1) - 2:{\~{N}at(?x-2)({7 6 1 0},{}),Plus(o, ?x-2, ?x-2)({5 4 3 2},{})}
(1 0) - 3:{~Plus(?x-5, ?x-4, ?x-3)({3 2 1 0},{}),
          Plus(suc(?x-5), ?x-4, suc(?x-3))({7 6 5 4},{})}
(1 0) - 4:{~Plus(suc(suc(o)), suc(o), ?x-6)({7 6 5 4},{3 2 1 0})}

dual-particles:
(1 1) - 0:{\~{N}at(?x-2)({2},{}),\~{N}at(?x-1)({1},{}),Nat(o)({0},{}),
          ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 1:{\~{N}at(?x-2)({2},{}),Nat(suc(?x-1))({1},{}),Nat(o)({0},{}),
```

```
            ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 2:{Plus(o, ?x-2, ?x-2)({2},{}),\~{N}at(?x-1)({1},{}),Nat(o)({0},{}),
            ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 3:{Plus(o, ?x-2, ?x-2)({2},{}),Nat(suc(?x-1))({1},{}),Nat(o)({0},{}),
            ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 4:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
            Plus(o, ?x-2, ?x-2)({2},{}),
            Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{}),
            Nat(o)({0},{}),Nat(suc(?x-1))({1},{})}
(1 1) - 5:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
            Plus(o, ?x-2, ?x-2)({2},{}),
            Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{}),Nat(o)({0},{}),
            \~{N}at(?x-1)({1},{})}
(1 1) - 6:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),\~{N}at(?x-2)({2},{}),
            Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{}),Nat(o)({0},{}),
            Nat(suc(?x-1))({1},{})}
(1 1) - 7:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),\~{N}at(?x-2)({2},{}),
            Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{}),Nat(o)({0},{}),
           \~{N}at(?x-1)({1},{})}
**********************************************************

Finding contradictions...

Enter theorem number: 4
==========================================================
```

A cada ciclo de inferência, é solicitado o número da cláusula relativa ao teorema que se deseja contradizer. Isto não é feito de forma automática para poder aumentar as possibilidades de teste da estratégia. Informado o número do teorema, as contradições deste são tratadas. Assim, duas novas teorias são geradas e "boxeadas" com a teoria original. Com esta, novo ciclo de inferência é iniciado:

```
**********************************************************
theory: New

particles:
(0 1) - 0:{Nat(o)({7 6 5 4 3 2 1 0},{})}
(0 1) - 1:{\~{N}at(?x-1)({7 5 2 0},{}),Nat(suc(?x-1))({6 4 3 1},{})}
```

```
(1 1) - 2:{\~{N}at(?x-2)({7 6 1 0},{}),Plus(o, ?x-2, ?x-2)({5 4 3 2},{})}
(1 0) - 3:{~Plus(?x-5, ?x-4, ?x-3)({7 6 5 4},{}),
           Plus(suc(?x-5), ?x-4, suc(?x-3))({3 2 1 0},{})}
(1 0) - 4:{~Plus(suc(suc(o)), suc(o), ?x-6)({3 2 1 0},{7 6 5 4})}
(1 0) - 5:{Plus(suc(o), o, suc(o))({7 6 5 4},{3 2 1 0})}
(1 0) - 6:{Plus(suc(o), suc(o), suc(suc(o)))({7 6 5 4},{3 2 1 0})}

dual-particles:
(1 1) - 0:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),\~{N}at(?x-2)({2},{}),
        Nat(o)({0},{}),\~{N}at(?x-1)({1},{}),
        Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{5 6})}
(1 1) - 1:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),\~{N}at(?x-2)({2},{}),
           Nat(o)({0},{}),Nat(suc(?x-1))({1},{}),
           Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{5 6})}
(1 1) - 2:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
           Plus(o, ?x-2, ?x-2)({2},{}),Nat(o)({0},{}),
          \~{N}at(?x-1)({1},{}),Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{5 6})}
(1 1) - 3:{~Plus(suc(suc(o)), suc(o), ?x-6)({4},{}),
           Plus(o, ?x-2, ?x-2)({2},{}),Nat(o)({0},{}),
           Nat(suc(?x-1))({1},{}),Plus(suc(?x-5), ?x-4, suc(?x-3))({3},{5 6})}
(1 1) - 4:{Plus(o, ?x-2, ?x-2)({2},{}),Nat(suc(?x-1))({1},{}),
           Nat(o)({0},{}),Plus(suc(o), o, suc(o))({5},{}),
           Plus(suc(o), suc(o), suc(suc(o)))({6},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 5:{Plus(o, ?x-2, ?x-2)({2},{}),\~{N}at(?x-1)({1},{}),
           Nat(o)({0},{}),Plus(suc(o), o, suc(o))({5},{}),
           Plus(suc(o), suc(o), suc(suc(o)))({6},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 6:{\~{N}at(?x-2)({2},{}),Nat(suc(?x-1))({1},{}),
           Nat(o)({0},{}),Plus(suc(o), o, suc(o))({5},{}),
           Plus(suc(o), suc(o), suc(suc(o)))({6},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
(1 1) - 7:{\~{N}at(?x-2)({2},{}),\~{N}at(?x-1)({1},{}),Nat(o)({0},{}),
           Plus(suc(o), o, suc(o))({5},{}),
           Plus(suc(o), suc(o), suc(suc(o)))({6},{}),
           ~Plus(?x-5, ?x-4, ?x-3)({3},{4})}
**********************************************************
```

```
Finding contradictions...

Enter theorem number: 4
=======================================================

Contradictions
0: {New (3 2 1 0) {(?x-6/suc(?x-3)), (?x-4/suc(o)), (?x-5/suc(o))}}
1: {New (6 1) {(?x-2/suc(?x-1))}}
2: {New (7 6 1 0) {(?x-2/o)}}
3: {New (7 5 2 0) {(?x-1/o)}}
4: {New (5 4) {(?x-3/?x-2), (?x-4/?x-2), (?x-5/o)}}
5: {New (7 6 5 4) {(?x-3/suc(suc(o))), (?x-4/suc(o)), (?x-5/suc(o))}}
6: {New (7 6 5 4) {(?x-3/suc(o)), (?x-4/o), (?x-5/suc(o))}}

Substitution: {(?x-6/suc(?x-3)), (?x-5/suc(o)), (?x-4/suc(o)), (?x-3/suc(suc(o)))
Answer: (~Plus(suc(suc(o)), suc(o), suc(suc(suc(o)))))
=======================================================
>
```

Novamente é solicitado o número do teorema. Ao ser tratadas as substituições a contradição da teoria é alcançada. A resposta é então fornecida. Com o uso desta estratégia percebe-se que, ao longo da prova, apenas duas cláusulas foram geradas.

Capítulo 5

Conclusão

5.1 O trabalho apresentado

O objetivo principal deste trabalho foi apresentar um método de inferência lógica denominado de *Método de Inferência baseado na Transformação Dual*, bem como um algoritmo para a realização da *transformação dual* propriamente dita.

Com este intuito, foi realizado um estudo dos método e algoritmo propostos a fim de possibilitar sua implementação. A implentação foi desenvolvida na linguagem de programação Common Lisp utilizando CLOS (do inglês, "Common Lisp Object System"), a extensão orientada objeto padrão do Common Lisp. O ambiente resultante desta implentação revelou-se uma importante ferramenta. Este ambiente vem sendo utilizado como auxílio para o estudo de potencialidades e restrições do método proposto. Estratégias para o método também têm sido testadas com o auxílio do ambiente desenvolvido.

Na apresentação deste trabalho preferiu-se passar a essência dos métodos e algoritmos propostos ao invés de apresentar suas definições formais. Para tal, ambos foram explicados passo a passo com exemplos e notas explicativas. Sua formalização foi deixada em segundo plano. De qualquer forma, o estudo desta é necessário para o leitor que deseja aprofundar o conhecimento de tais método e algoritmo. A apresentação deste trabalho deu-se na seguinte seqüência:

- Foi apresentada uma introdução geral abordando a Inteligência Artificial (IA), sua origem e grandes áreas. Destas, foi dada ênfase à IA Simbólica, por ser a grande área de abrangência deste trabalho. Foi apresentado também o papel da lógica e dos métodos de inferência na IA.

- Seguiu-se com a apresentação dos mecanismos dedutivos da lógica. Foi dada, primeiramente uma noção de inferência lógica, seguindo-se para métodos de prova

dedutivos e de refutação, passando-se aos métodos de prova automática (por computador). Destes foram discutidas algumas características importantes para o entendimento do trabalho. Por fim, o método proposto (objeto do trabalho) foi apresentado. Este método possui uma interessante característica: não se utiliza de uma regra de inferência externa ao sistema lógico, para a inferência apenas é explorada a dualidade das possíveis representações de uma fórmula lógica e suas semânticas. Na definição do método, foi priorizado o enfoque didático a fim de facilitar o entendimento do mesmo, sua dinâmica e propriedades.

- Foi discutido o problema da transformação dual (um dos passos "caros" de método proposto). Também objeto do trabalho, o algoritmo proposto foi apresentado. Para tal, foi dada uma noção informal da dinâmica do algoritmo antes de sua definição ser apresentada, o que foi feito logo após. O algoritmo proposto realiza, de forma eficiente, a transformação de uma fórmula lógica entre as formas normais canônicas conjuntiva e disjuntiva. O algoritmo possui duas importantes propriedades: gera diretamente o conjunto de implicantes primários de um conjunto de cláusulas ou cláusulas duais sem gerar cláusulas (duais) subsumidas, e, devido à sua estrutura o mesmo é naturalmente concorrente e, por isto, facilmente paralelizável. A descrição deste algoritmo também encontra-se em [TB97].

- O ambiente que implementa método e algoritmo proposto foi apresentado. O mesmo é um ambiente computacional para inferência lógica de primeira ordem que utiliza uma estrutura de dados na forma de um hipercubo como área de trabalho para procedimentos outros, além dos já definidos neste trabalho, como eliminação de cláusulas subsumidas de um conjunto de cláusulas, combinação de duas ou mais fórmulas já representadas em forma normal, etc. Porém, a concorrência citada foi apenas simulada. Nocões gerais válidas para todo o ambiente foram abordadas e suas principais primitivas de interface foram apresentadas. Exemplos de execuções foram mostrados e comentados. O ambiente implementado faz parte do ambiente apresentado em [TB96]

5.2 PERSPECTIVAS

- Extensão do algoritmo de transformação dual para lógicas não clássicas. O algoritmo de transformação dual proposto foi originalmente definido com base em uma estrutura matemática genérica. Embora o algoritmo e sua implementação tenham sido descritos em termos da linguagem da lógica de primeira ordem, a transformação dual se aplica a outras estruturas matemáticas. Em particular, o

algoritmo proposto é correto para qualquer estrutura que contenha ao menos a estrutura $\Lambda = (\mathcal{F}, \oplus, \otimes)$, onde $(\mathcal{F}, \oplus)$ e $(\mathcal{F}, \otimes)$ são dois semi-grupos comutativos interrelacionados por axiomas distributivos. Por exemplo, formalismos como a lógica nebulosa [Zad79], a lógica de possibilidades [DP88] e lógicas de mais de dois valores verdades [Bel77] satisfazem esta restrição.

- Definição de estratégias de prova para o método de inferência proposto. Estudos e testes promissores têm sido realizados neste campo. Uma destas, especificamente, tem apresentado bons resultados: baseia-se na definição de metas (*goals*) a serem cumpridas a cada ciclo de inferência. A meta atualmente testada é a priorização da contradição da cláusula que representa o teorema negado. A estrutura desta estratégia lembra a estrutura adotada na linguagem de programação PROLOG [SS86].

- Estudo mais aprofundado das propriedades do método proposto. Por exemplo, as substituições associadas a subsunções de literais não são tratadas pelo método. Outras propriedades lógicas talvez possam ser aplicadas no tratamento simultâneo de uma fórmula lógica em suas duas formas de representação.

- Comparação do método de inferência proposto com outros métodos encontrados na literatura.

- Dar continuidade ao estudo do projeto de um sistema híbrido para modelagem cognitiva integrando os enfoques conexionista e simbólico descrito em [Bit97b], do qual este trabalho faz parte.

- Implementação concorrente do algoritmo de transformação dual de acordo com a proposta descrita em [BT97].

Referências Bibliográficas

[Bel77] N.D. Belnap. A useful four-valued logic. In J.M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logics*. D. Reidel Pub. Co., 1977.

[Bit96a] G. Bittencourt. Boxing theories (abstract). *Workshop on Logic, Language,Information and Computation (WoLLIC'96), Journal of the Interest Group in Pure and Applied Logics (IGPL), Imperial College, London and Max-Planck-Institut, Saarbrücken*, 4(1):479–481, 1996. ISSN 0945-9103, Held in Salvador, Brasil, May 8-10.

[Bit96b] G. Bittencourt. *Inteligência Artificial - Ferramentas e Teorias*. Décima Escola de Computação, 1996. 8 a 13 de julho, Campinas, SP.

[Bit97a] G. Bittencourt. Concurrent inference through dual transformation. *Journal of the Interest Group in Pure and Applied Logics (IGPL), in press*, 1997.

[Bit97b] G. Bittencourt. In the quest of the missing link. In *Proceedings of IJCAI 15, Nagoya, Japan, August 23-29*, pages 310–315. Morgan Kaufmann (ISBN 1-55860-480-4), 1997.

[BT97] G. Bittencourt and I. Tonin. A multi-agent approach to first-order logic. In *Proceedings of 8^{th} Portuguese Conference on Artificial Intelligence (EPIA '97), Coimbra, Portugal, October 6-9*, pages 167–178. Springer-Verlag (ISBN 3-540-63586-6), 1997.

[CF91] W.-T. Chen and M.-Y. Fang. An efficient procedure for theorem proving in propositional logic on vector computers. *Parallel Computing*, 17:983–5, 1991.

[CRT73] C.-L. Chang and Lee R.C.-T. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Computer Science Classics, 1973.

[Dah83] V. Dahl. Logic programming as representation of knowledge. *IEEE Computer*, pages 106–111, October 1983.

[DP88] D. Dubois and H. Prade. *Possibility Theory - An Approach to the Computerized Processing of Uncertainty.* Plenum Press, 1988.

[Eis86] N. Eisinger. What you always wanted to know about clause graph resolution. In *Proceedings of the 8th ICAD, Springer-Verlag LNCS No. 230*, pages 316–336, 1986.

[Fit90] M. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, New York, 1990.

[GN87] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers Inc, Los Altos, California, 1987.

[Isr94] David J. Israel. The role(s) of logic in artificial intelligence. In C.J.Hogger Dov M.Gabbay and J.A.Robinson, editors, *Handbook of Logic in Artificil Intelligence and Logic Programing - Logical Foundations*, volume 1. Oxford University Press, 1994.

[LM86] R. Loganantharaj and R.A. Mueller. Parallel theorem proving with connection graphs. In *Proceedings of the 8th ICAD, Springer-Verlag LNCS No. 230*, pages 337–352, 1986.

[MV88] T.R. Martinez and J.J. Vidal. Adaptative parallel logic networks. *Journal of Parallel and Distributed Computing*, 5(1):26–58, February 1988.

[Nil82] N.J Nilsson. *Principles of Artificial Intelligence.* Springer-Verlag, Berlin, Heidelberg, New York, 1982.

[Pen91] Roger Penrose. *A Mente Nova do Rei - computadores, mentes e as leis da física.* Campus, 1991.

[Qui59] W.V.O. Quine. On cores and prime implicants of truth functions. *American Mathematics Monthly*, 66:755–760, 1959.

[Qui87] M.J. Quinn. *Desingning Efficient Algorithms for Parallel Computers.* McGraw-Hill International Editions, 1987.

[RN95] S.J. Russell and P. Norvig. *Artificial Intelligence A Modern Approach.* Prentice Hall, Upper Saddle River, NJ, 1995.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[Smu68] R.M. Smullyan. *First Order Logic.* Springer-Verlag, 1968.

[Soc91] R. Socher. Optimizing the clausal normal form transformation. *Journal of Automated Reasoning*, 7(3):325–336, 1991.

[SS86] L. Sterling and E. Shapiro. *The Art of Prolog.* M.I.T. Press Series in Logic Programming, Cambridge, MA, London, England, 1986.

[TB96] I. Tonin and G. Bittencourt. Logik: Um ambiente para ensino de lógica. In *V Congresso Íbero-Americano de Educação Superior em Computação EDUC'96, Cidade do México, 19 a 21 de setembro*, 1996.

[TB97] I. Tonin and G. Bittencourt. Uma implementação orientada a objetos para a transformação dual em lógica de primeira ordem. In *Proceedings of XIV Brazilian Software and Hardware Seminars - SEMISH 97, Brasília, 2 a 8 agosto*, pages 411–422, 1997.

[WV94] Larry Wos and Robert Veroff. Logical basis for automating reasoning. In C.J.Hogger Dov M.Gabbay and J.A.Robinson, editors, *Handbook of Logic in Artificil Intelligence and Logic Programing - Deduction Methodologies*, volume 2. Oxford University Press, 1994.

[Zad79] L.A. Zadeh. A theory of approximate reasoning. In D. Mitchie J.E. Hayes and L.I. Mikulich, editors, *Machine Intelligence 9.* Wiley Masson, New York, 1979.